HP Design Center
Training Course

HP 64000-UX Overview Course

Student Workbook

Course No. HP 64120T

HEWLETT PACKARD

APPLICATION SUPPORT DIVISION
100 Mayfield Ave., Mountain View, CA 94043

64120-90050 Printed in U.S.A. 09/87

Copyright

Table of Contents

Course Synopsis - 64120T

☐

INTRODUCTION FOR STUDENTS

A brief synopsis of the 64000-UX Overview Course is given below:

Course Name:	HP 64000-UX Overview Course
Course Number:	64120T
Objective:	Provide the student with the basics of the 64000-UX system. The student should gain enough practical information through the course labs to "come up to speed" on the 64000-UX system and emulation.
Content:	An overview of the 64000-UX system including: access to the computer, the User Interface Software (pmon),the Softkey Driven Editor (Sk), command files, the 64000-UX language tools, concepts of measurement systems, hardware option_test, philosophy of emulation, and emulation based on the Motorola 68000.
Format:	Lecture (45%) / Lab (50%) / Overhead (5%)
Duration:	2 days
Number of Students:	8 MAXIMUM
Products Covered:	64120A, 64801S, 64808S, 64790S, 64243S, 64302A, 64815S, 64819S, 64845S
Prerequisite:	No prerequisite required
Location:	HP sales office or Training Center
Options:	None

☐

INTRODUCTION FOR STUDENTS

A brief synopsis of the 64000-UX Overview Course is given below:

Course Name:	HP 64000-UX Overview Course
Course Number:	64120X
Objective:	Provide the student with the basics of the 64000-UX system. The student should gain enough practical information through the course labs to "come up to speed" on the 64000-UX system and emulation.
Content:	An overview of the 64000-UX system including: access to the computer, the User Interface Software (pmon),the Softkey Driven Editor (Sk), command files, the 64000-UX language tools, concepts of measurement systems, hardware option_test, philosophy of emulation, and emulation based on the Motorola 68000.
Format:	Lecture (45%) / Lab (50%) / Overhead (5%)
Duration:	2 days
Number of Students:	8 MAXIMUM
Products Covered:	64120A, 64801S, 64808S, 64790S, 64243S, 64302A, 64815S, 64819S, 64845S
Prerequisite:	No prerequisite required
Location:	Customer Site
Options:	64120X #001 - Add One Student On-site

Course Schedule

☐

SUGGESTED COURSE SCHEDULE:

The schedule shown below is only a suggested schedule. The actual amount of time spent on each section will have to be determined by the instructor depending upon the customer's needs. The amount of on-hands lab time should not be altered, or the effectiveness of the training course will be compromised.

HP 64000-UX Training Course **Day 1**

8:00-8:10	(1)	Course Introduction
8:10-8:20	(2)	System Introduction
8:20-8:50	(3)	User Interface Software (pmon)
8:50-8:55	(4)	HELP!!
8:55-9:25	(5)	System Introduction Lab
9:25-9:35		Break
9:35-10:05	(6)	Editors
10:05-10:35	(7)	Command Files
10:35-12:00	(8)	Editor / Command Files Lab
12:00-1:00		Lunch
1:00-1:10	(9)	HP 64000-UX Language System
1:10-1:40	(10)	Assemblers
1:40-2:10	(11)	Compilers
2:10-2:20		Break
2:20-2:50	(12)	Linkers
2:50-3:00	(13)	Language System Summary
3:00-5:00	(14)	Assemblers / Compilers / Linkers Lab

HP 64000-UX Training Course **Day 2**

8:00-9:00	(15)	Measurement Systems
9:00-9:30	(16)	Measurement Systems Lab
9:30-9:40		Break
9:40-11:30	(17)	Introduction to Emulation
11:30-12:30		Lunch
12:30-2:00	(18)	Emulation Lab
2:00-2:10		Break
2:10-4:30	(18)	Emulation Lab
4:30-4:45	(19)	The 64000-UX Product Line
4:45-5:00	(20)	Course Conclusion

☐

COURSE OUTLINE (64120T):

HP 64000-UX Overview Course **Day 1**

- (a) Course Introduction
  - (1) general information
    - (i) who should attend
    - (ii) philosophy of the course
    - (iii) course objectives
  - (2) knowledge survey
    - (i) purpose and use of the survey
    - (ii) complete the survey

- (b) System Introduction
  - (1) software development process
    - (i) idea
    - (ii) specify/design phase
    - (iii) implement/construct phase
    - (iv) test/debug phase
    - (v) product
  - (2) where does Logic Systems Division fit in?
    - (i) specify/design phase
      - * Teamwork SA/SD/RT
    - (ii) implement/construct phase
      - * Sk and Vi editors
      - * assemblers/linkers
      - * C/Pascal compilers
      - * ADA connection
    - (iii) debug/test phase
      - * emulators
      - * timing/state analyzers
      - * software performance analyzer
      - * software analyzers
- (c) User Interface Software (pmon)
  - (1) purpose
  - (2) features / advantages / benefits
  - (3) getting started
    - (i) entering pmon
    - (ii) special features
      - * control characters
      - * hidden commands
    - (iii) softkeys
      - * definition
      - * terminal keys
      - * default pmon screen
        - the display screen
        - the STATUS line
        - the command lines
        - the softkeys

HP 64000-UX Overview Course (cont) **Day 1**

(iv) HP-UX command mapping
- * HP-UX /pmon cross referencing
- * entering HP-UX commands
- * gaining access to the shell

(v) exiting pmon

(4) command interpretation

(5) last look at the manual

(d) HELP!!

(1) Where is help available?

(i) HP-UX help
- * on-line manuals for HP-UX commands
- * the HP-UX system manuals
- * help within HP-UX applications: using "?"

(ii) HP 64000-UX help
- * on-line manuals for HP 64000-UX commands
- * the HP 64000-UX system manuals
- * help within HP 64000-UX applications: "HELP" softkey

(2) This training course as a source of information

(e) System Introduction Lab

(f) Editors

(1) What is available?

(i) the HP-UX system visual editor: Vi

(ii) the HP 64000-UX system full-screen editor: Sk

(2) a brief look at Vi

(i) features / advantages / benefits

(ii) drawbacks / limitations

(iii) shipped with HP-UX

(3) an in-depth look at Sk

(i) features / advantage / benefits

(ii) drawbacks / limitations

(iii) ordering information

(iv) modes of the editor
- * command mode
- * insert mode
- * revise mode

(v) the editor screen
- * format
  - text area
  - line numbers
- * status line
- * command line
- * softkeys

☐

HP 64000-UX Overview Course (cont) Day 1

- (vi) cursor and screen control
- (vii) special keys sequences
  - * recalling commands from the command buffer
  - * refreshing the screen
  - * entering non-printable characters
- (viii) commands of the editor
  - * common commands available
  - * unique and advanced commands
- (ix) accessing the editor
  - * entering the editor
  - * exiting the editor

(g) Command Files
- (1) definition of command files
  - (i) features / advantages / benefits of command files
- (2) what command files are available
  - (i) HP-UX command files: shell scripts
    - * creating shell scripts
    - * using shell scripts
    - * parameter passing
    - * example shell script: the ".profile" file
  - (ii) HP 64000-UX command files
    - * creating these command files
    - * using these command files
    - * parameter passing
    - * possible problems using HP 64000-UX command files
      - - entering errors while creating command files
    - * example of pmon command file
    - * example of Sk command file

(i) HP 64000-UX Language System
- (1) features / advantages / benefits
- (2) components of the language system
- (3) source code
- (4) absolute code

(j) Assemblers
- (1) purpose
- (2) features / advantages / benefits
- (3) functional components
  - (i) initialization
  - (ii) pass1
  - (iii) pass2
  - (iv) pass3
  - (V) pass4

☐

HP 64000-UX Overview Course (cont) **Day1**

- (4) processor directives
  - (i) uses and syntax
- (5) output files of the assembler
  - (i) relocatable file (.R)
  - (ii) symbol file (.A)
  - (iii) listing file (.O)
    - * options available
      - - expansion
      - - cross-referencing
- (6) assembler directives
  - (i) function
  - (ii) most useful directives
    - * ORG
    - * END
- (7) transfer address
  - (i) definition
  - (ii) creation of the transfer address

(k) Compilers

- (1) purpose of a compiler
- (2) features / advantages / benefits
- (3) functional description
  - (i) optional preprocessor
  - (ii) pass1
  - (iii) pass2
  - (iv) pass3
- (4) types of compilers
  - (i) native compilers
  - (ii) cross compilers
- (5) compiler libraries
- (6) output files of a compiler
  - (i) relocatable file (.R)
  - (ii) symbol file (.A)
  - (iii) listing file (.O)
    - * options available
      - - expansion
      - - cross-referencing
  - (iv) assembly file (ASM68000)
- (7) processor directives
  - (i) purpose
  - (ii) syntax for C
  - (iii) syntax for Pascal

☐

HP 64000-UX Overview Course (cont) **Day1**

(8) transfer address
  (i) overview
  (ii) creation of the transfer address in C
  (iii) creation of the transfer address in Pascal

(l) Linkers
  (1) purpose of a linker
  (2) features / advantages / benefits
  (3) functional components
    (i) initialization
    (ii) pass1
    (iii) pass2
    (iv) cross reference
  (4) output files of a linker
    (i) absolute file (.X)
    (ii) linker symbol file (.L)
    (iii) linker listing file (.O, .MAP files)
      * generation and syntax
      * options
        - cross-referencing
    (iv) linker command file (.K)
      * features / advantages / benefits
      * creation
        - queries of the linker
        - ASCII file generation

(m) Language System Summary
  (1) file types
    (i) source files
    (ii) relocatable files
    (iii) assembler / compiler symbol files
    (iv) assembler / compiler listing files
    (v) linker symbol file
    (vi) linker listing file
    (vii) linker command file
    (viii) object code or absolute file
  (2) flow diagram of language system

(n) Assemblers / Compilers / Linkers Lab

☐

HP 64000-UX Overview Course **Day 2**

(a) Measurement Systems
  (1) manual references
    (i) HP 64000-UX User's Guide
        HP P/N 64801-90903
  (2) definition of a module
    (i) available 64000-UX modules
  (3) definition of a measurement system
    (i) IMB connection between modules
    (ii) definition of IMB
  (4) building measurement systems
    (i) msinit
      * searching for cardcages
    (ii) msconfig
    (iii) msstat
      * status messages
        - available
        - unusable
        - running
        - locked
    (iv) msunlock
  (5) hardware option test
    (i) look at 64120
      * front panel
      * back panel
    (ii) look at module (emulation)
      * boards in module
      * boards configuration in cardcage
      * option test software
(b) Measurement Systems Lab
(c) Introduction to Emulation
  (1) what is emulation (definition)
  (2) emulation vs. simulation
    (i) real time execution
    (ii) non-real time execution
  (3) what is transparency
    (i) functional transparency
    (ii) timing transparency
    (iii) electrical transparency
  (4) what is an emulation monitor
    (i) foreground monitors
    (ii) background monitors
  (5) block diagram of emulation
    (i) connection between cards and diagram
    (ii) connection between cables and buses
    (iii) connection between hardware and cardcage

☐

HP 64000-UX Overview Course (cont) **Day 2**

(6) components of emulation
- (i) program loading
- (ii) configuring the emulator
  - * default configuration
  - * modify configuration
  - * clock specification
  - * restriction to real time runs
  - * illegal opcode detection
  - * memory mapping
    - - break on write to ROM
    - - default conditions
    - - processor specific questions
    - - kinds of memory
      - - ROM
      - - RAM
      - - guarded
    - - sources of memory
      - - emulation
      - - target
    - - memory overlays
  - * pod configuration
  - * interactive measurements
  - * configuration command files
- (iii) run control
  - * run [from] / [until]
  - * single stepping
  - * break on trigger condition
  - * software breakpoints
  - * reset (user and system)
- (iv) display / modify commands
  - * registers
  - * memory
  - * symbols
  - * I/O ports
- (v) edbuild
- (vi) using analysis with emulation
  - * state / timing / spa
  - * internal analyzer (64302)
    - - use with the emulator
    - - setting trace specifications
    - - starting and stopping trace
    - - displaying trace data

☐

HP 64000-UX Overview Course (cont) **Day 2**

(vii) simulated I/O (brief discussion of existence)
- * definition of simulated I/O
- * standard input
- * standard output
- * standard error
- * performing simulated I/O
  - - keyboard
  - - file (ex. device files)
- * displaying simulated I/O

(d) Emulation Lab

(e) The 64000-UX Product Line
- (1) overall picture of the 64000-UX product line
- (2) what training is available for the HP 64000-UX options
  - (i) overview of training available from LSD

(f) Course Conclusion
- (1) course conclusions
  - (i) information the student should know
  - (ii) what the student should be able to do with HP 64000-UX
- (2) knowledge survey
  - (i) purpose and use of the survey
  - (ii) complete the survey

Course Evaluations

☐

COURSE EVALUATIONS BY STUDENTS AND INSTRUCTORS:

Two types of evaluations for the course need to be filled out by the students. The first is the standard course evaluation form supplied by the Application Support Division (ASD) and is read by an op-scan machine. The instructor is also required to fill out an evaluation form to be returned to ASD. The front page of this evaluation sheet is for general information about the student and an overview of the knowledge level of the student in a few general areas. This front page is to be filled out before beginning the course. The back page of the evaluation contains generic questions concerning the training course presentation, labs, materials, and instructor and should be completed after finishing the course.

The second type of evaluation sheet is supplied with the course and is meant to be filled out once before the course, and once after the course is complete. The second evaluation or "knowledge survey" will be used to determine where the weak points of the course are so that the revisions to the training course can reflect the needs of the students. The first of the two knowledge surveys should be completed prior to beginning the course and the second copy of the knowledge survey should be completed after finishing the training course.

All the evaluations should be collected by the instructor and returned to Application Support Division for review.

Knowledge Survey - Course Preview

☐

Course Preview - HP 64000-UX Overview Course - 64120T

Before beginning the training course, take a few minutes to indicate how you the student feel about performing each of the tasks listed below. Use the table below as a guide to determine what you feel your confidence level is for each of the areas specified. Circle your choice and return the completed form to the course instructor.

EC - Extremely Confident
C - Confident
SC - Somewhat Confident
NS - Not Sure
U - Unable

Your are now able to:

1. Login and logoff an HP 9000 series 300 computer.
   Degree of Confidence: EC C SC NS U

2. Access the user interface software, "pmon", and use it effectively.
   Degree of Confidence: EC C SC NS U

3. Gain on-line help and help contained in the 64000-UX system manuals.
   Degree of Confidence: EC C SC NS U

4. Use a text editor such as Sk or Vi.
   Degree of Confidence: EC C SC NS U

5. Create and use command files effectively.
   Degree of Confidence: EC C SC NS U

6. Assemble and compile source code using the 64000-UX Language System.
   Degree of Confidence: EC C SC NS U

7. Link the output of the Assemblers and Compilers to form absolute code.
   Degree of Confidence: EC C SC NS U

8 . Build and modify measurement systems.
   Degree of Confidence: EC C SC NS U

9 . Understand the philosophy of emulation.
   Degree of Confidence: EC C SC NS U

10. Access emulation and have the ability to use the basic commands.
   Degree of Confidence: EC C SC NS U

11. Understand the functionality of the internal analyzer and its use with the emulator.
   Degree of Confidence: EC C SC NS U

☐

Course Review - HP 64000-UX Overview Course - 64120T

After finishing the training course, take a few minutes to indicate how you the student feel about performing each of the tasks listed below. Use the table below as a guide to determine what you feel your confidence level is for each of the areas specified. Circle your choice and return the completed form to the course instructor.

EC - Extremely Confident
C - Confident
SC - Somewhat Confident
NS - Not Sure
U - Unable

Your are now able to:

1. Login and logoff an HP 9000 series 300 computer.
   Degree of Confidence: EC C SC NS U

2. Access the user interface software, "pmon", and use it effectively.
   Degree of Confidence: EC C SC NS U

3. Gain on-line help and help contained in the 64000-UX system manuals.
   Degree of Confidence: EC C SC NS U

4. Use a text editor such as Sk or Vi.
   Degree of Confidence: EC C SC NS U

5. Create and use command files effectively.
   Degree of Confidence: EC C SC NS U

6. Assemble and compile source code using the 64000-UX Language System.
   Degree of Confidence: EC C SC NS U

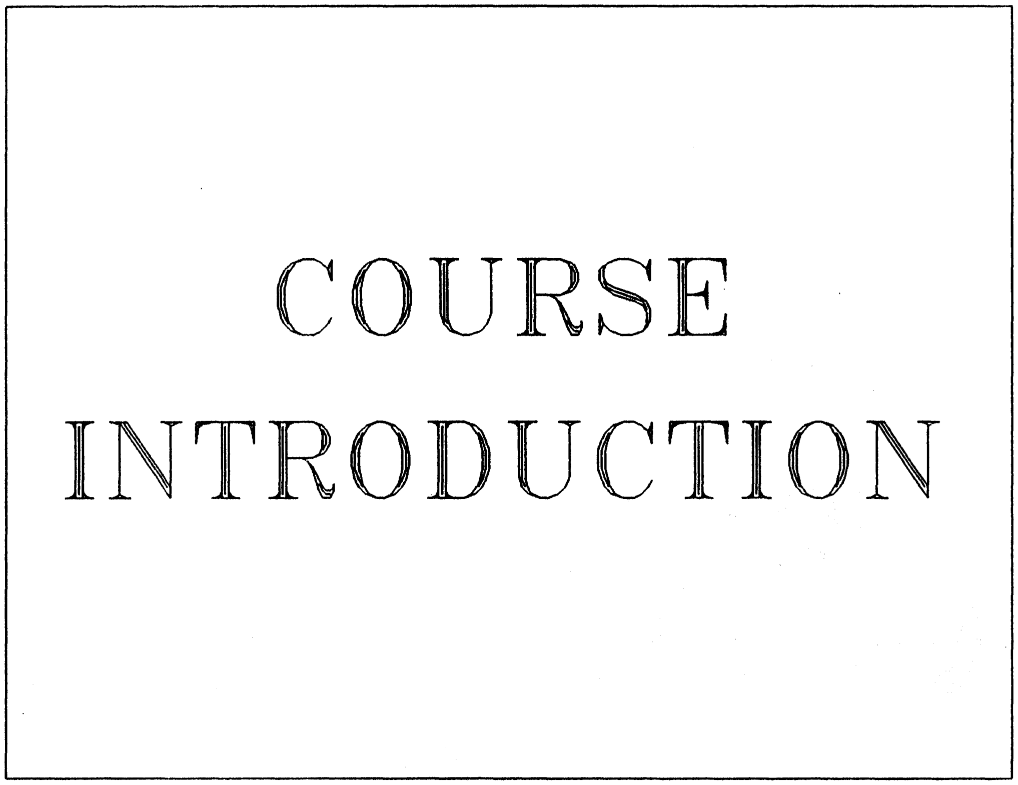
7. Link the output of the Assemblers and Compilers to form absolute code.
   Degree of Confidence: EC C SC NS U

8. Build and modify measurement systems.
   Degree of Confidence: EC C SC NS U

9. Understand the philosophy of emulation.
   Degree of Confidence: EC C SC NS U

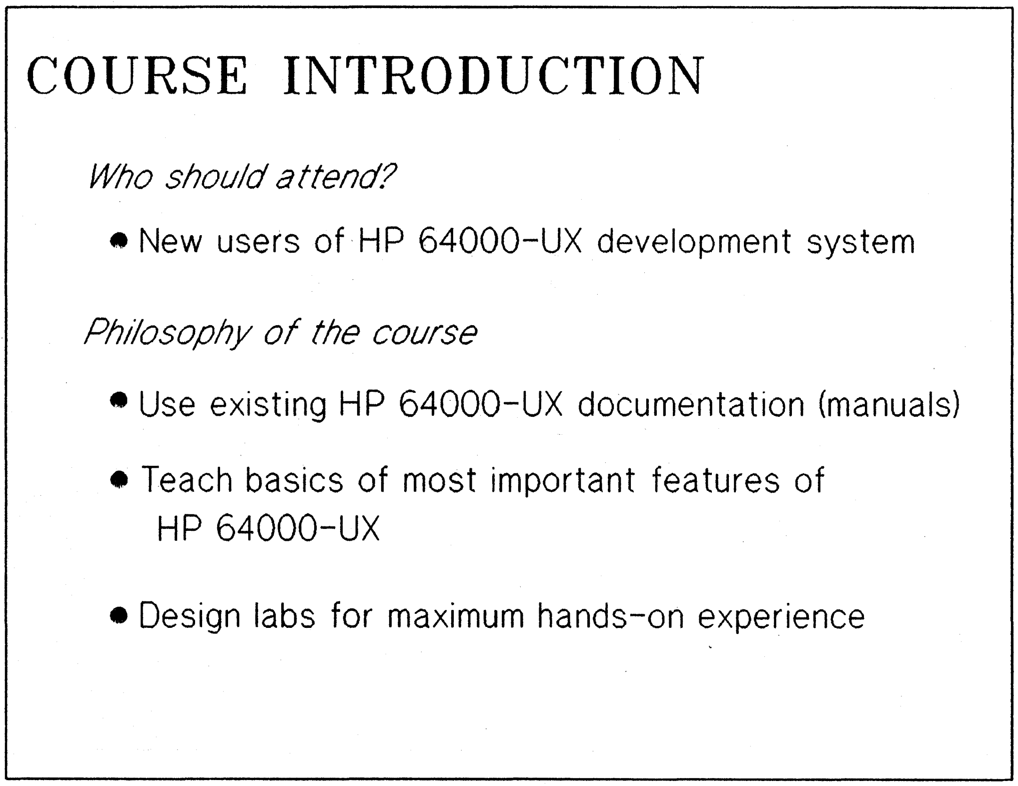
10. Access emulation and have the ability to use the basic commands.
    Degree of Confidence: EC C SC NS U

11. Understand the functionality of the internal analyzer and its use with the emulator.
    Degree of Confidence: EC C SC NS U



COURSE INTRODUCTION

CII 1.05



COURSE INTRODUCTION

Who should attend?

- New users of HP 64000-UX development system

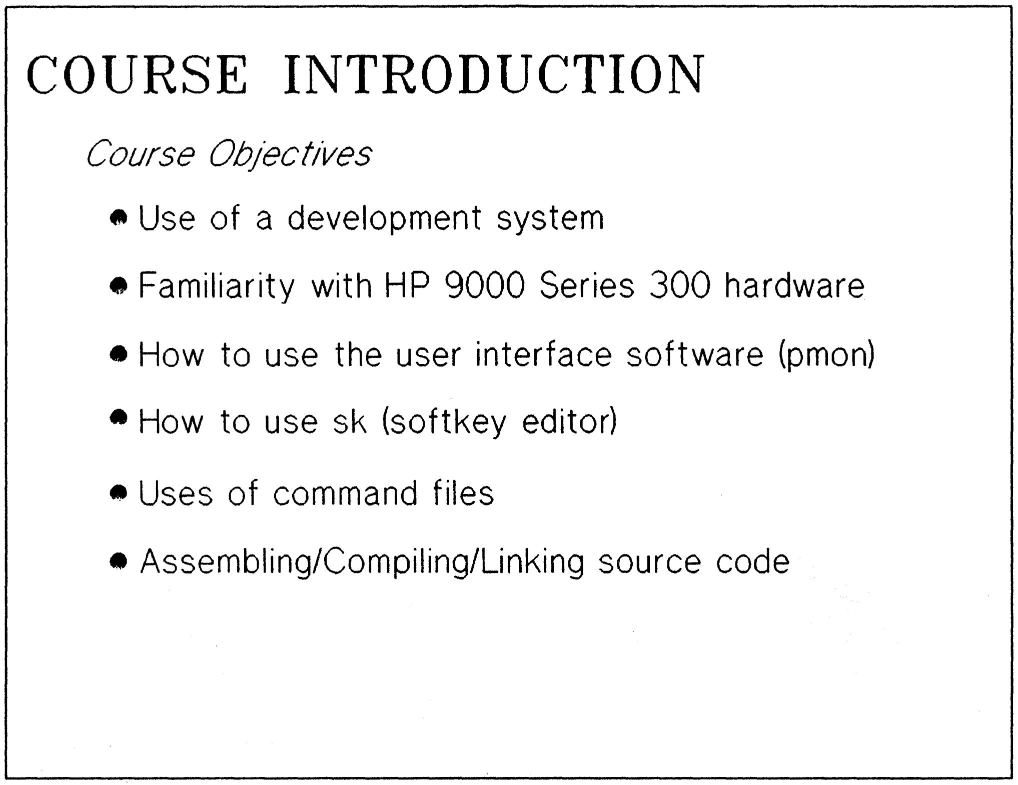
Philosophy of the course

- Use existing HP 64000-UX documentation (manuals)
- Teach basics of most important features of HP 64000-UX
- Design labs for maximum hands-on experience

CII 1.10

STUDENT NOTES:

Course Introduction



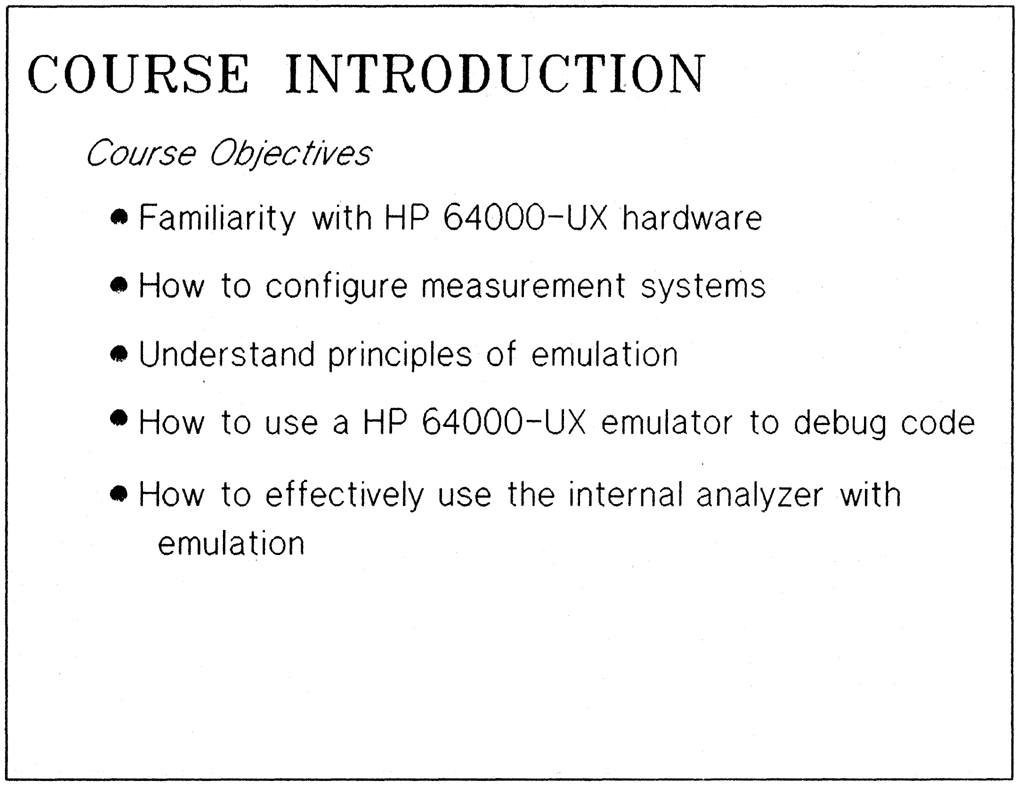
COURSE INTRODUCTION

Course Objectives

- Use of a development system
- Familiarity with HP 9000 Series 300 hardware
- How to use the user interface software (pmon)
- How to use sk (softkey editor)
- Uses of command files
- Assembling/Compiling/Linking source code

CII 1.15 © 1987 Hewlett-Packard Company

STUDENT NOTES:



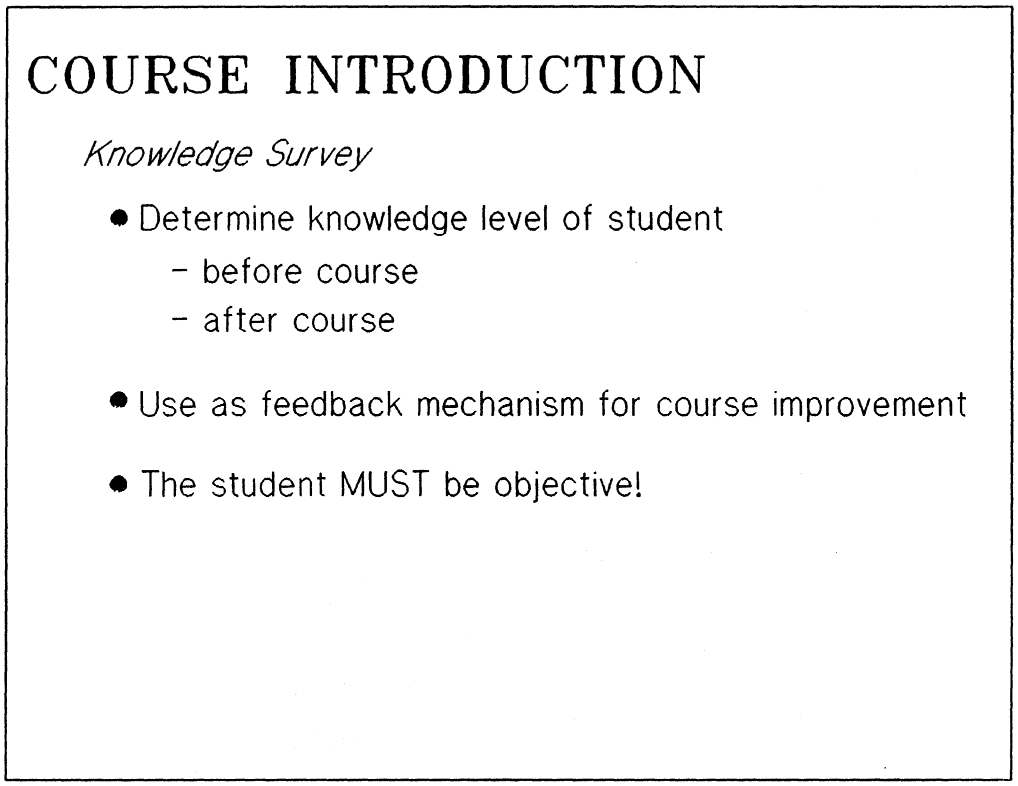
COURSE INTRODUCTION

Course Objectives

- Familiarity with HP 64000-UX hardware
- How to configure measurement systems
- Understand principles of emulation
- How to use a HP 64000-UX emulator to debug code
- How to effectively use the internal analyzer with emulation

CII 1.20

STUDENT NOTES:



COURSE INTRODUCTION

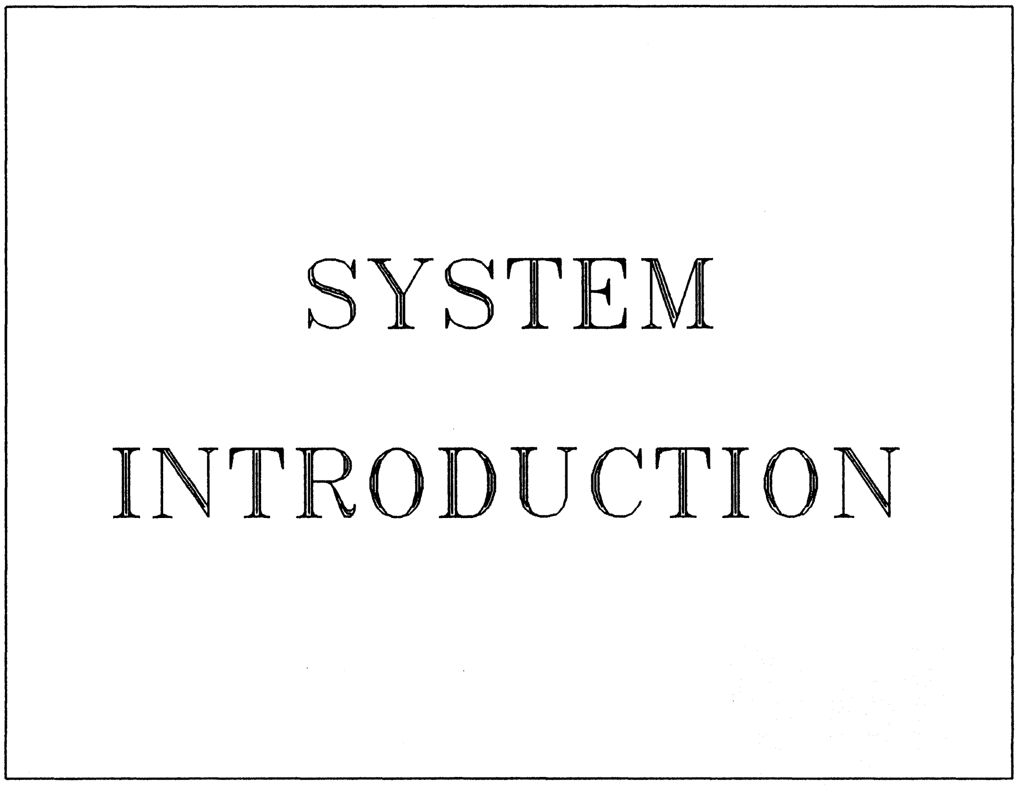
Knowledge Survey

- Determine knowledge level of student
  - before course
  - after course
- Use as feedback mechanism for course improvement
- The student MUST be objective!

CII 1.25 ○ 1987 Hewlett-Packard Company

STUDENT NOTES:

☐ System Introduction



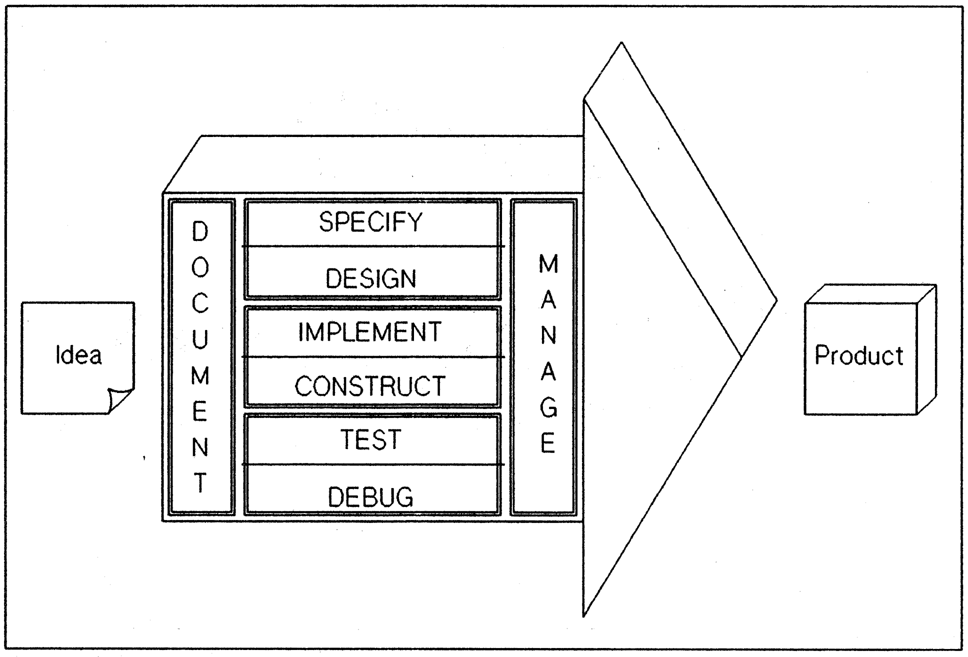
SYSTEM

INTRODUCTION

SI1 1.05

☐ System Introduction

Software Development Process



SI10115 © 1987 Hewlett-Packard Company

NOTES:

The diagram above gives a general synopsis of the software development process of a project. The HP 64000-UX product line and HP-UX help the designer with the following parts of the design:

Specify/Design
Implement/Construct
Test/Debug
Manage

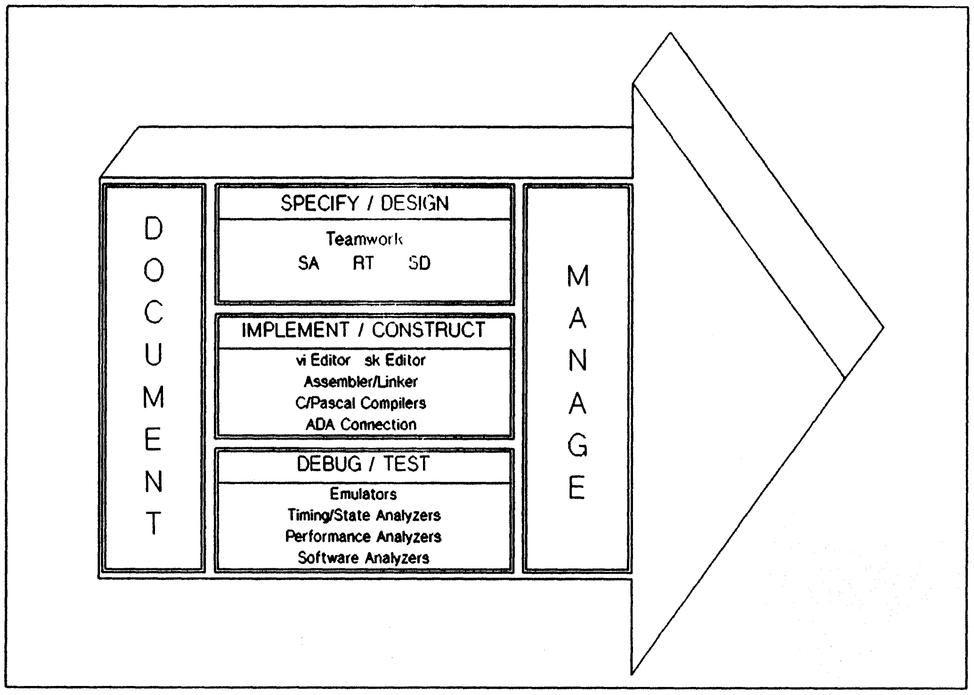
Hewlett-Packard currently sells a product called "Alis" which is used for project documentation and would aide with the following part of the design:

Document

STUDENT NOTES:

System Introduction

Software Development Solutions



SI10120 © 1987 Hewlett-Packard Company

NOTES:

The "Specify/Design" phase of a project can be handled with the Teamwork SA/SD/RT products.

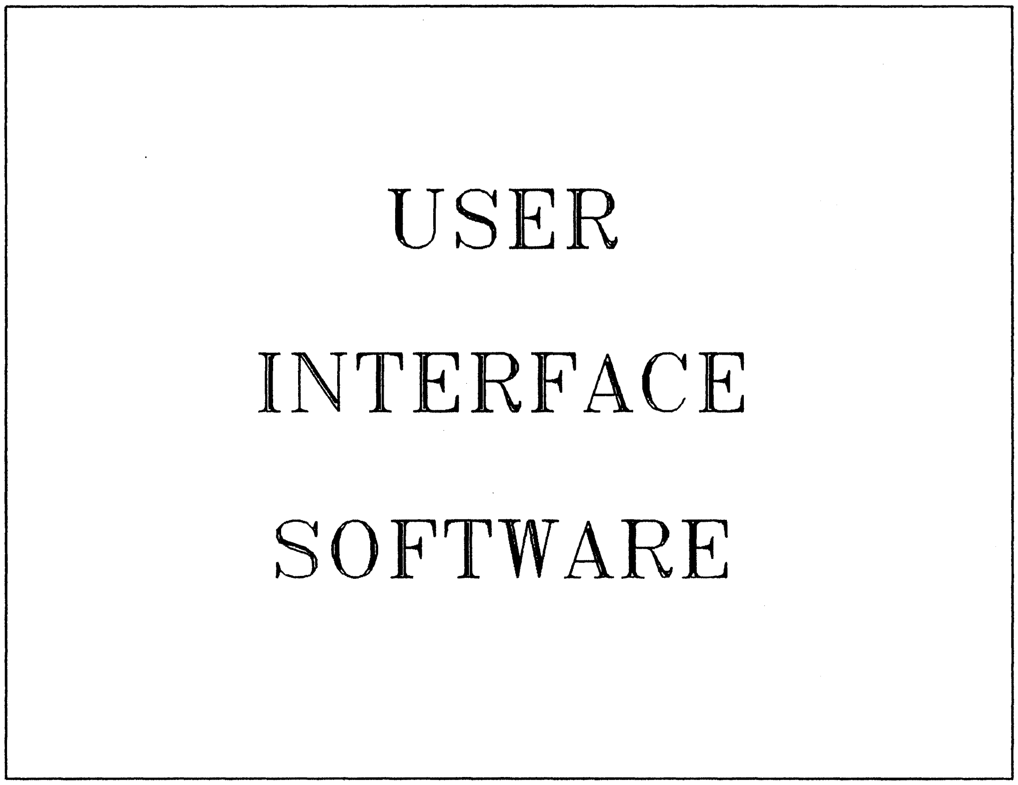
SA: Structured Analysis *SD:* Structured Design *RT:* Real-Time Extensions

The "Implement/Construct" phase can be handled with the Sk or Vi source code editors, the 64000-UX Language System, or the available ADA language tools.

The "Debug/Test" phase is were HP 64000-UX fits in. Logic Systems Division offers a wide variety of emulators and analyzers for almost every development need.

The "Manage" portion of a project can be handled with the utilities shipped with HP-UX, such as "SCCS" (source code control system) and "make" (intelligent software management tool).

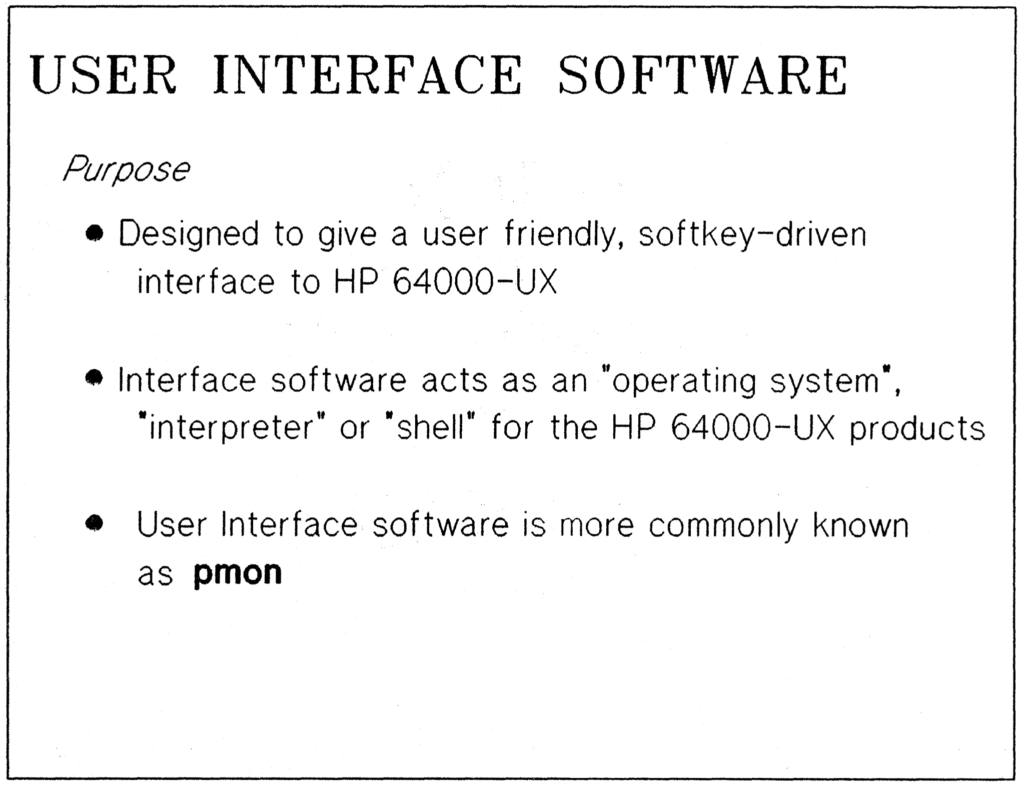
STUDENT NOTES:



USER INTERFACE SOFTWARE

IS1 1.05

□ Interface Software



USER INTERFACE SOFTWARE

Purpose

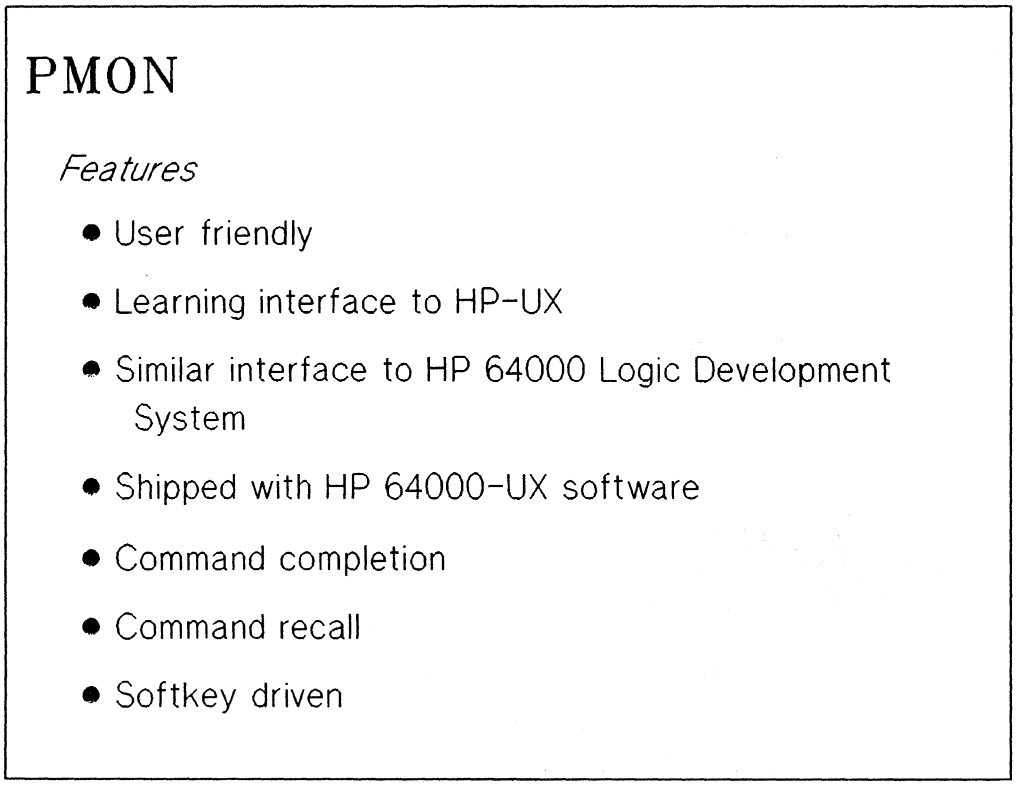
- Designed to give a user friendly, softkey-driven interface to HP 64000-UX
- Interface software acts as an "operating system", "interpreter" or "shell" for the HP 64000-UX products
- User Interface software is more commonly known as **pmon**

IS1 1.10 © 1987 Hewlett-Packard Company

NOTES:

Pmon is considered an "operating system" because it is the interpreter for some of the 64000-UX commands.

STUDENT NOTES:



PMON

Features

- User friendly
- Learning interface to HP-UX
- Similar interface to HP 64000 Logic Development System
- Shipped with HP 64000-UX software
- Command completion
- Command recall
- Softkey driven

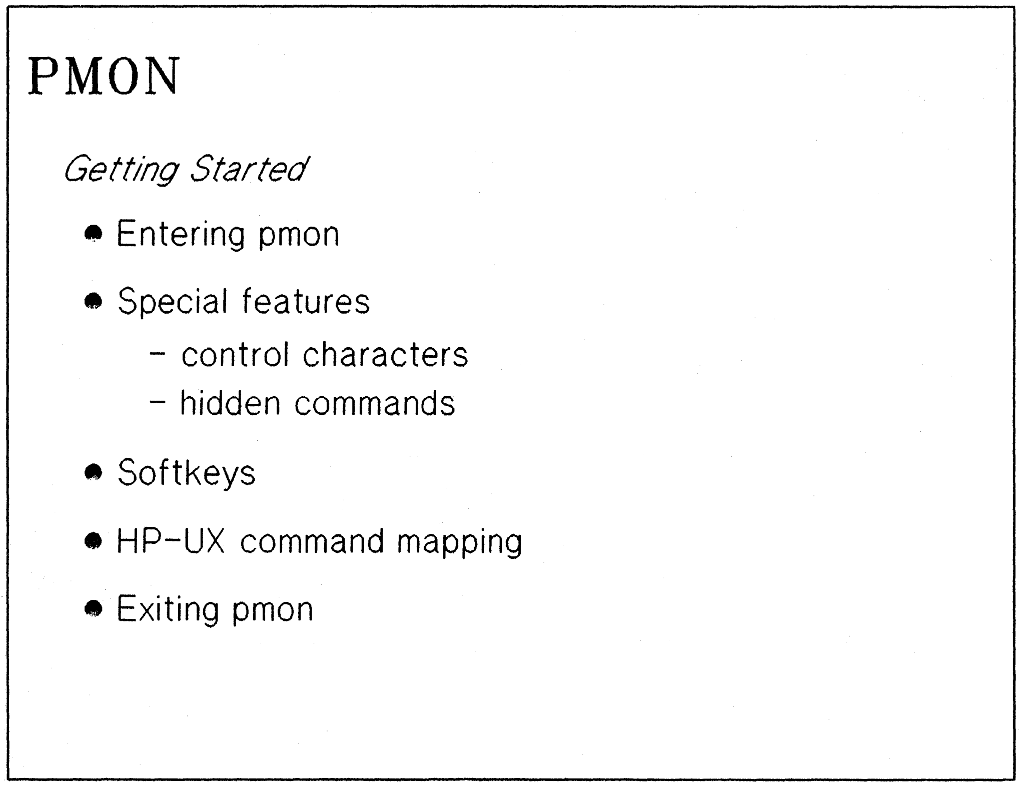
IS1 1.15 © 1987 Hewlett-Packard Company

MANUAL: page 1-2 User Interface Software Operating Manual

NOTES:

Command Completion- is accomplish by typing the first few letters of a command, pressing the tab key, and having pmon complete the command. This is an excellent time saving feature for the experienced user.

STUDENT NOTES:



PMON

Getting Started

- Entering pmon
- Special features
  - control characters
  - hidden commands
- Softkeys
- HP-UX command mapping
- Exiting pmon

IS1 1.20

MANUAL: page 1-4 thru 1-8 User Interface Software Operating Manual

NOTES:

Many control characters are available as time saving features of the command entry editor. A brief list of the most common commands is on page 1-4. A more complete list is in Appendix B of the manual.

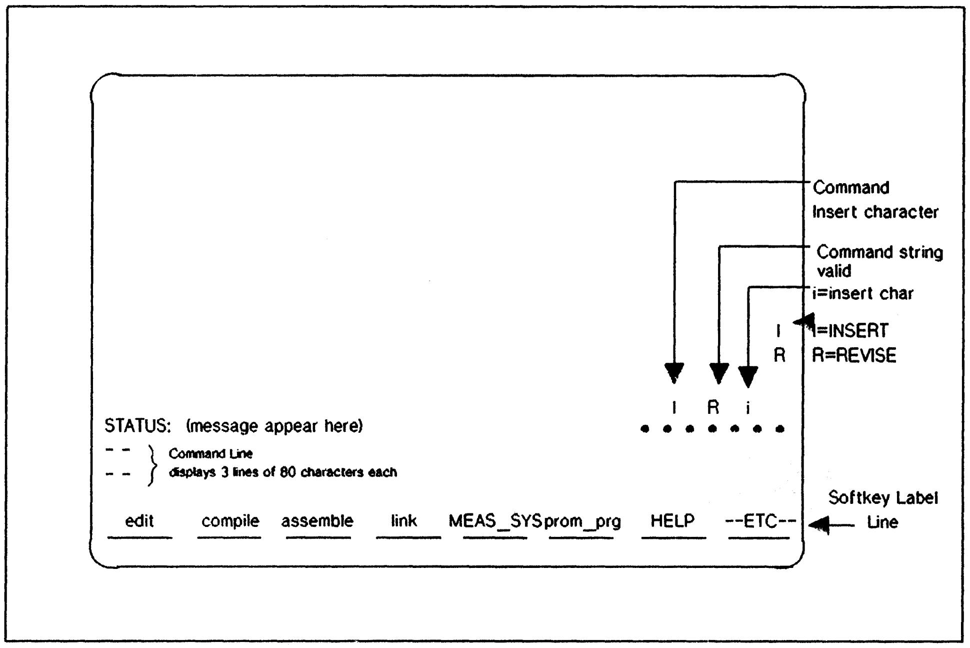
Some of these special commands are "hidden"; they have no softkey representation. The more useful commands are listed on page 1-8. A complete list of these commands is on page B-2 of Appendix B.

Softkeys: a physical representation of an HP-X command associated with one of the eight function keys (f1-f8) of the terminal keyboard. In essence, they map directly to HP-UX commands and options. A complete listing of the command mapping is contained in Appendix A.

STUDENT NOTES:

□ Interface Software

PMON



IS1 1.30 © 1986 Hewlett-Packard Company

NOTES:

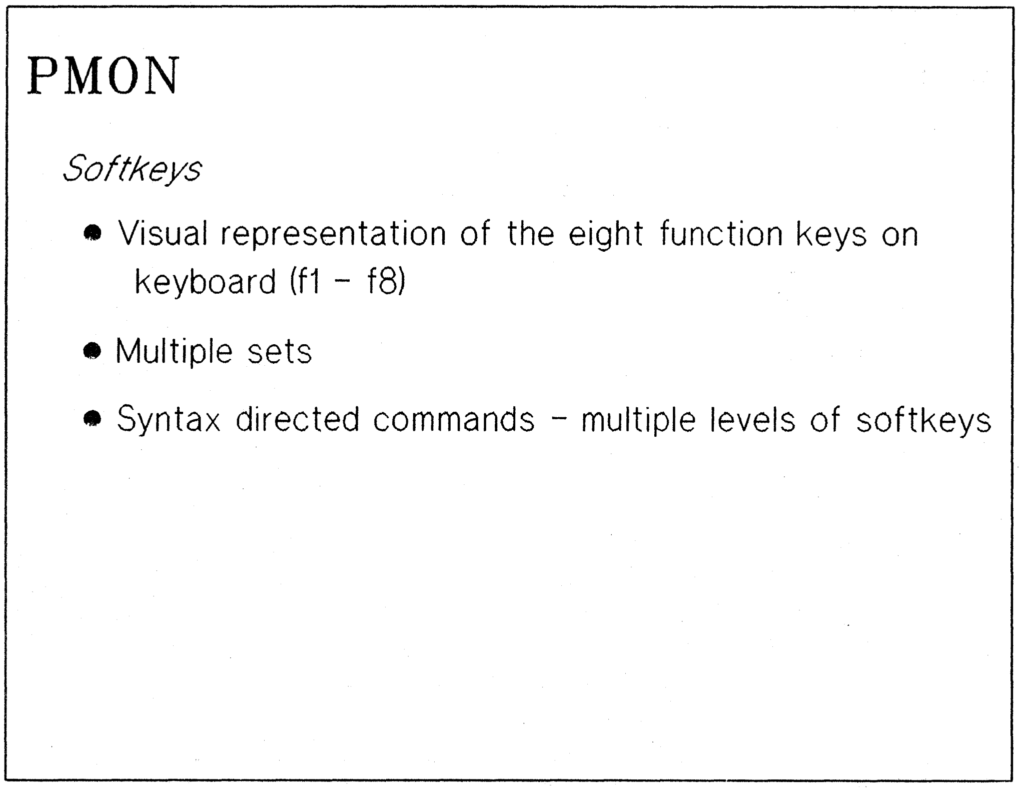
Command line: allows for commands up to three lines (240 characters) long

Status line: gives error messages as well as command information and explanations

User area: display are for output of commands

STUDENT NOTES:

☐ Interface Software



PMON

Softkeys

- Visual representation of the eight function keys on keyboard (f1 - f8)
- Multiple sets
- Syntax directed commands - multiple levels of softkeys

IS1 1.25 ○ 1987 Hewlett-Packard Company

NOTES:

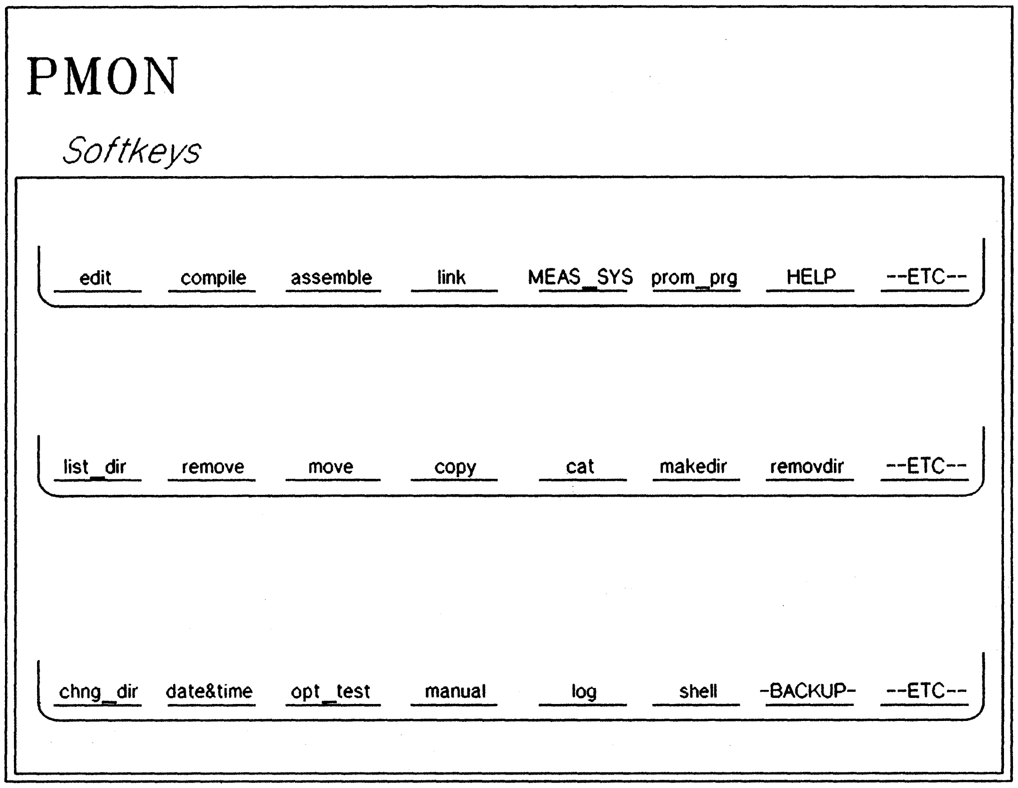
Eight function keys (f1-f8) are available on HP terminals such as the HP2392.

Multiple sets: many different sets of softkeys can be accessed with the "---ETC--" softkey.

Directed syntax: softkey labels change to aid the user in entering commands with the proper syntax.

STUDENT NOTES:

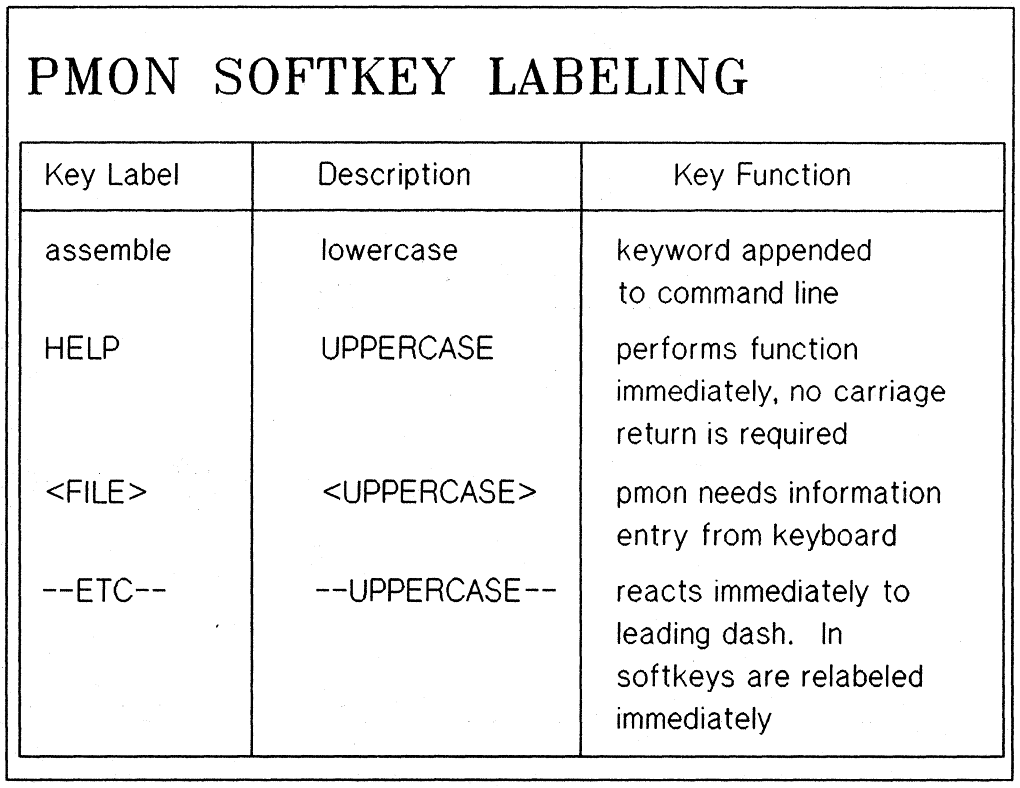
Interface Software



NOTES:

The three sets of softkeys for pmon are shown in the diagram above.

STUDENT NOTES:



PMON SOFTKEY LABELING

Key Label	Description	Key Function
assemble	lowercase	keyword appended to command line
HELP	UPPERCASE	performs function immediately, no carriage return is required
<FILE>	<UPPERCASE>	pmon needs information entry from keyboard
--ETC--	--UPPERCASE--	reacts immediately to leading dash. In softkeys are relabeled immediately

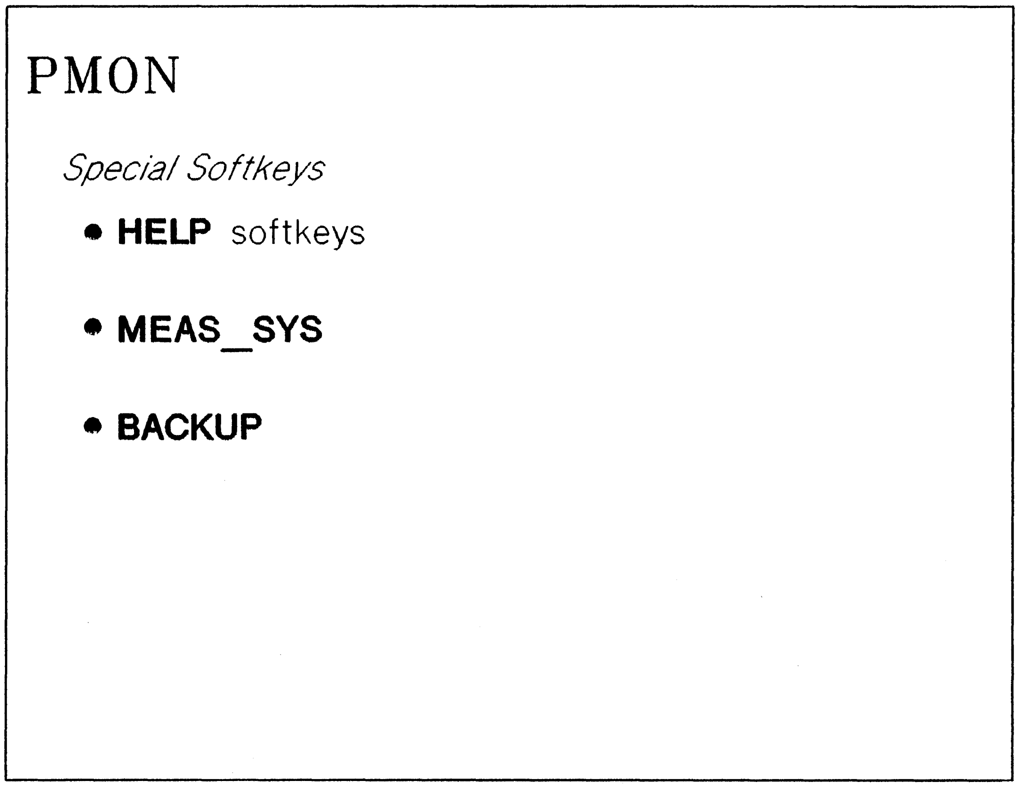
NOTES:

The table in the diagram above shows the possible types of commands which are available in pmon. Take note of these, especially the last one which requires user input.

Softkeys with the syntax: "<LABEL>" are not executable, but will give information and/or syntax for the information needed.

STUDENT NOTES:

Interface Software



PMON

Special Softkeys

- **HELP** softkeys
- **MEAS_SYS**
- **BACKUP**

IS1 1.35 © 1987 Hewlett-Packard Company

MANUAL: pages 1-6 thru 1-7 User Interface Software Operating Manual

NOTES:

HELP softkeys: appear at many levels and give brief descriptions of commands available at that time (page 1-6)

MEAS_SYS softkeys: are the interface to HP 64000-UX emulation hardware, analysis hardware and software (page 1-6). These features will be explained later (day 3).

BACKUP softkeys: allow the user to backup to tape drive or flexible disc.

STUDENT NOTES:

□ Interface Software

PMON

Directed Syntax

- Softkeys (**f1** - **f8**) dynamically relabeled
- Labels on display indicate function
- Keys relabeled as command entered or as cursor moves through command
- All valid keywords displayed

IS1 1.37 © 1987 Hewlett-Packard Company

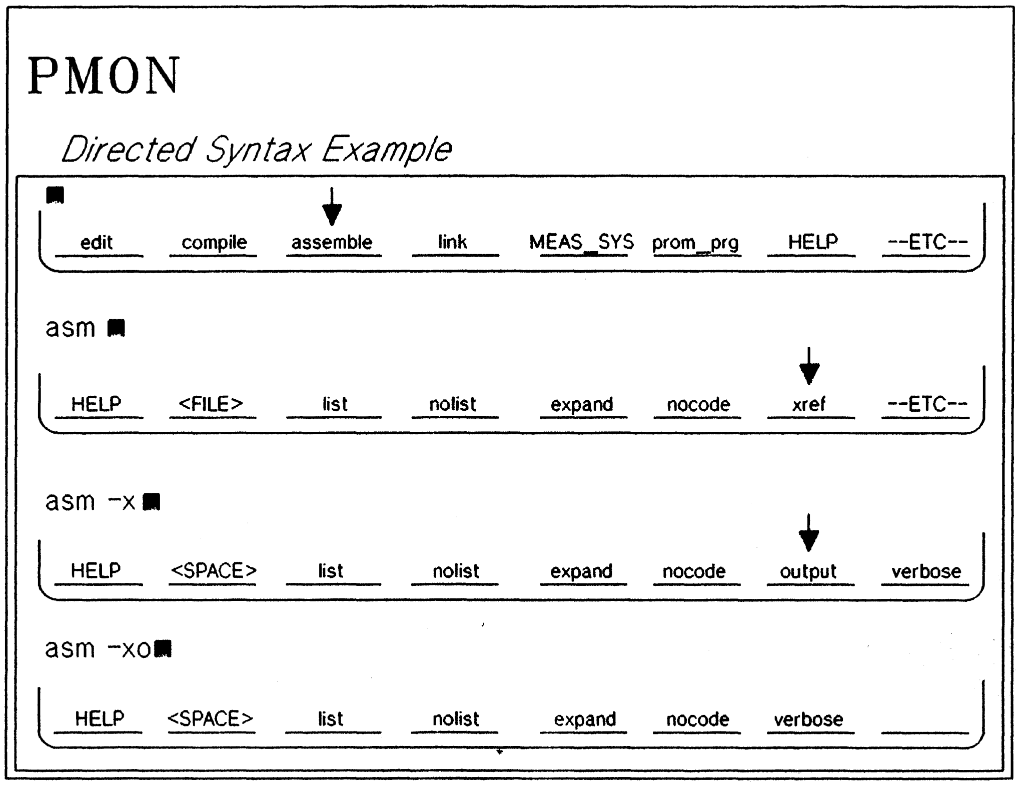
NOTES:

directed syntax: this allows commands to be entered quickly and effectively with little or no worries of making syntax errors.

The softkeys are relabeled every time another softkey is pushed, thus giving the user the choice of all the available options to a command.

STUDENT NOTES:

Interface Software



PMON

Directed Syntax Example

ISI 1.38 © 1987 Hewlett-Packard Company

NOTES:

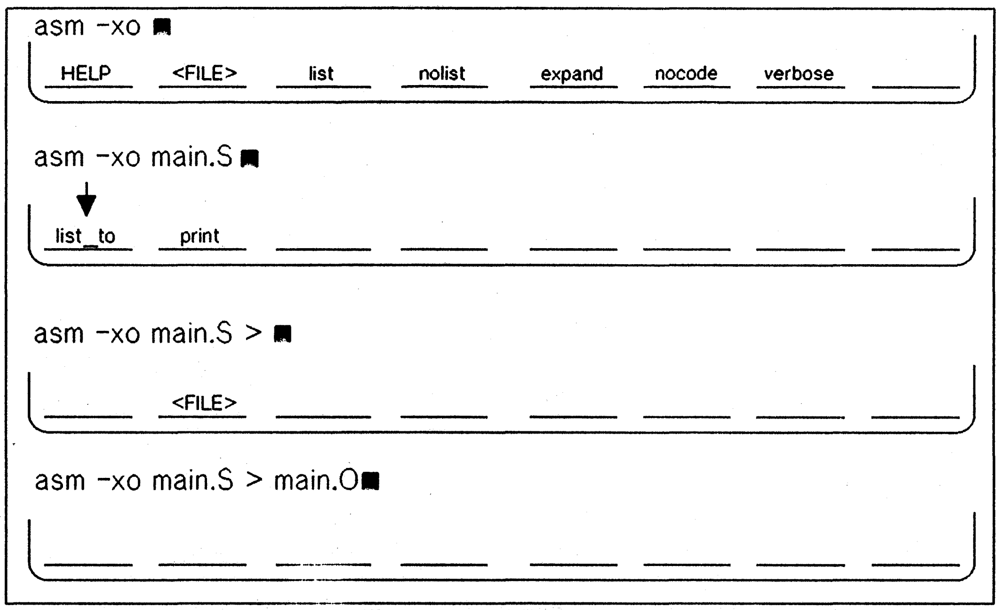
The next two slides show an example command being entered in pmon and how the softkeys change while the command is being built.

STUDENT NOTES:

Interface Software

PMON

Directed Syntax Example



```
asm -xo ■
```

```
asm -xo main.S ■
```

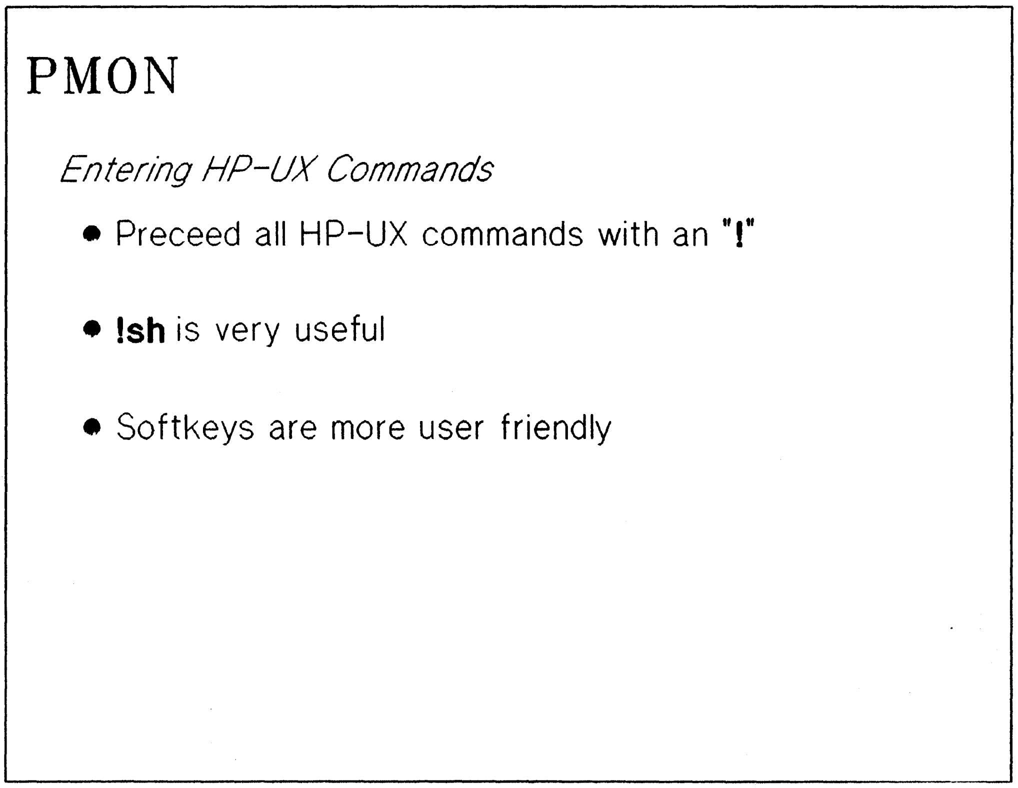
```
asm -xo main.S > ■
```

```
asm -xo main.S > main.O■
```

ISI 1.39 © 1987 Hewlett-Packard Company

STUDENT NOTES:

☐ Interface Software



PMON

Entering HP-UX Commands

- Preceed all HP-UX commands with an "**!**"
- **!sh** is very useful
- Softkeys are more user friendly

ISI 1.40 © 1987 Hewlett-Packard Company

MANUAL: page 1-7 User Interface Software Operating Manual

NOTES:

The "!" syntax is needed only while in pmon. It is not the syntax for normal HP-UX commands.

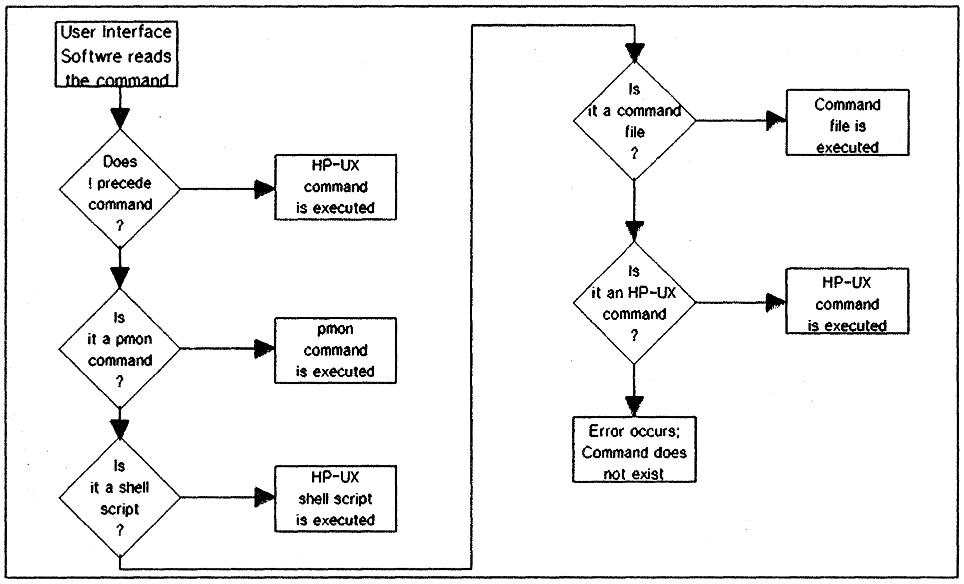
Softkeys are set-up so the user has a more friendly, learning interface to HP-UX commands.

STUDENT NOTES:

☐ Interface Software

PMON

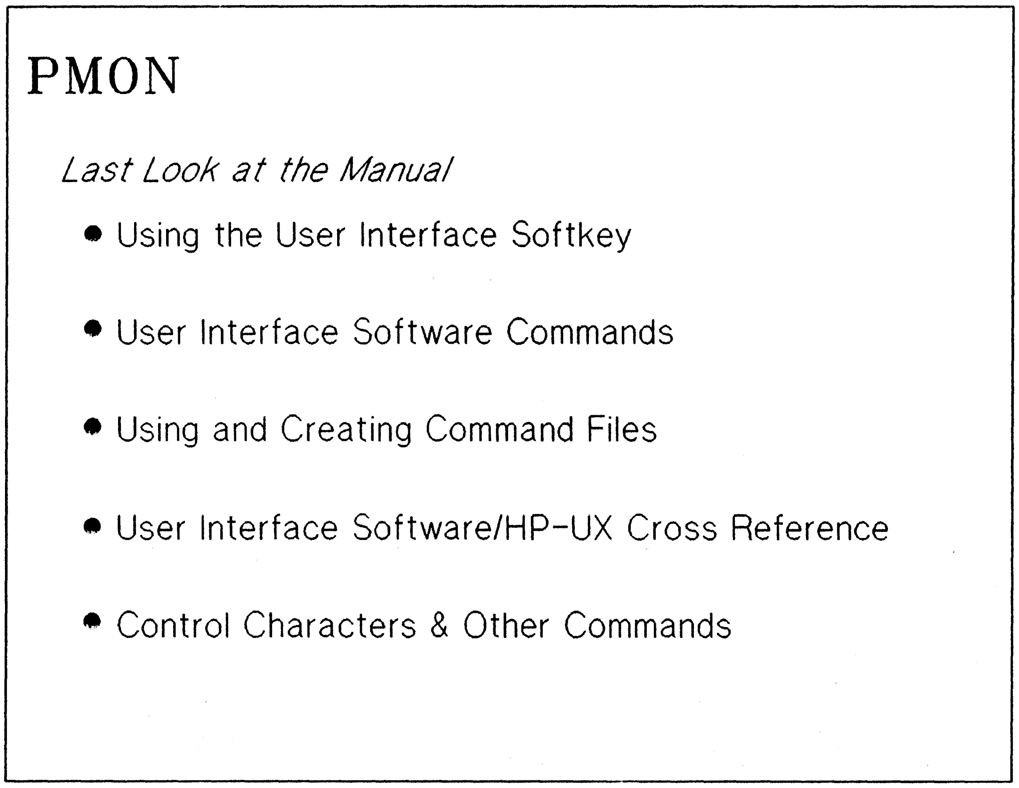
How Commands are Interpreted



IS1 1.45 ○ 1987 Hewlett-Packard Company

MANUAL: page 1-3 User Interface Software Operating Manual

STUDENT NOTES:



PMON

Last Look at the Manual

- Using the User Interface Softkey
- User Interface Software Commands
- Using and Creating Command Files
- User Interface Software/HP-UX Cross Reference
- Control Characters & Other Commands

ISI 1.50

NOTES:

Chapter 1 - general information
Chapter 2 - command syntax options
Chapter 3 - commands files will be discussed later today in their own section
Appendix A - HP-UX command cross referencing
Appendix B - more good information about special pmon commands

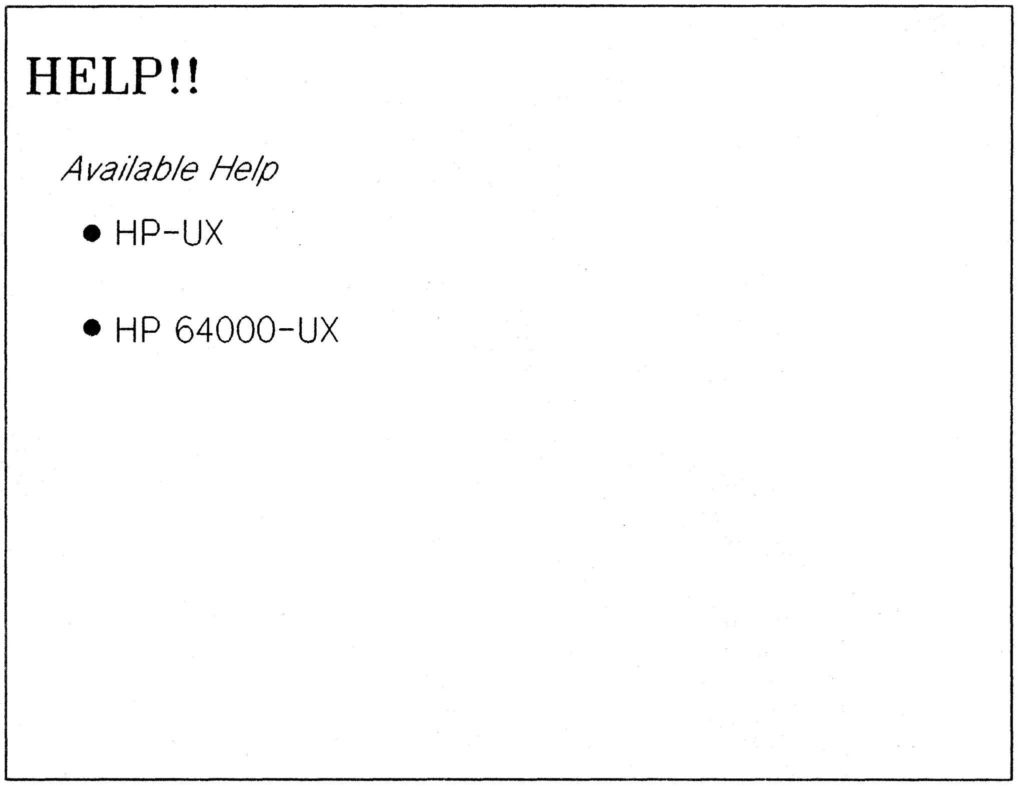
STUDENT NOTES:

☐ HELP!!

HELP!!

OV1 4.05 ○ 1987 Hewlett-Packard Company

☐ HELP!!



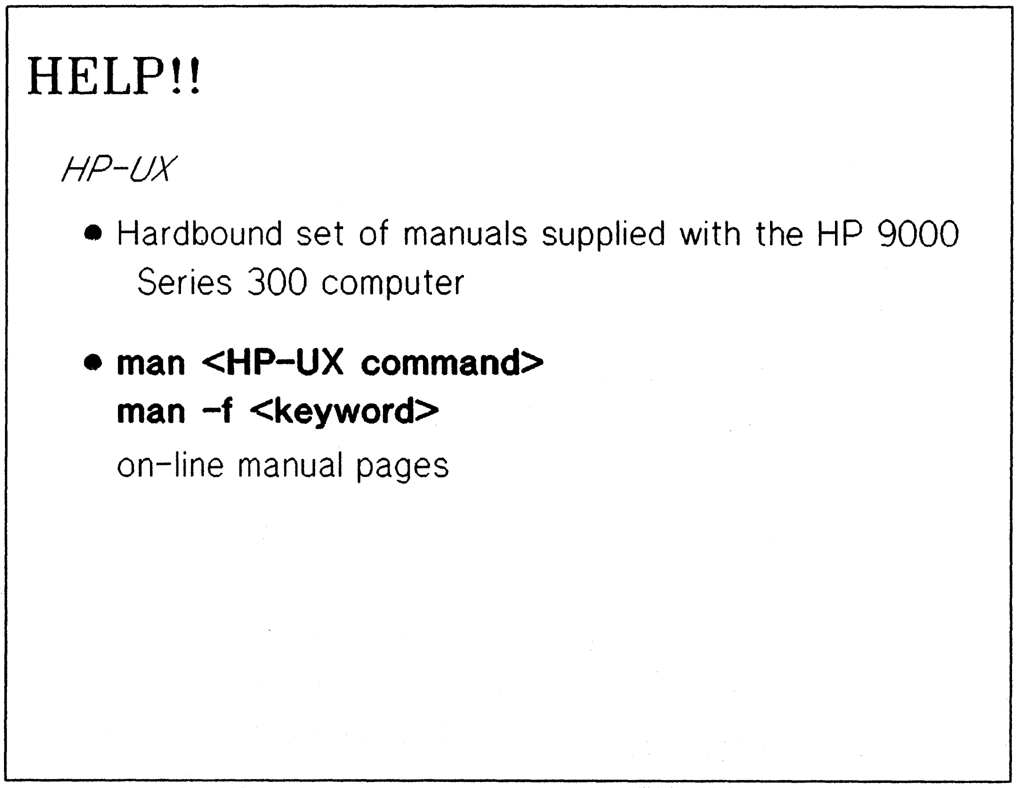
HELP!!

Available Help

- HP-UX
- HP 64000-UX

OVI 4.10 © 1987 Hewlett-Packard Company

☐ HELP!!



HELP!!

HP-UX

- Hardbound set of manuals supplied with the HP 9000 Series 300 computer
- **man <HP-UX command>**
  man -f <keyword>
  on-line manual pages

OVI 4.15 © 1987 Hewlett-Packard Company

NOTES:

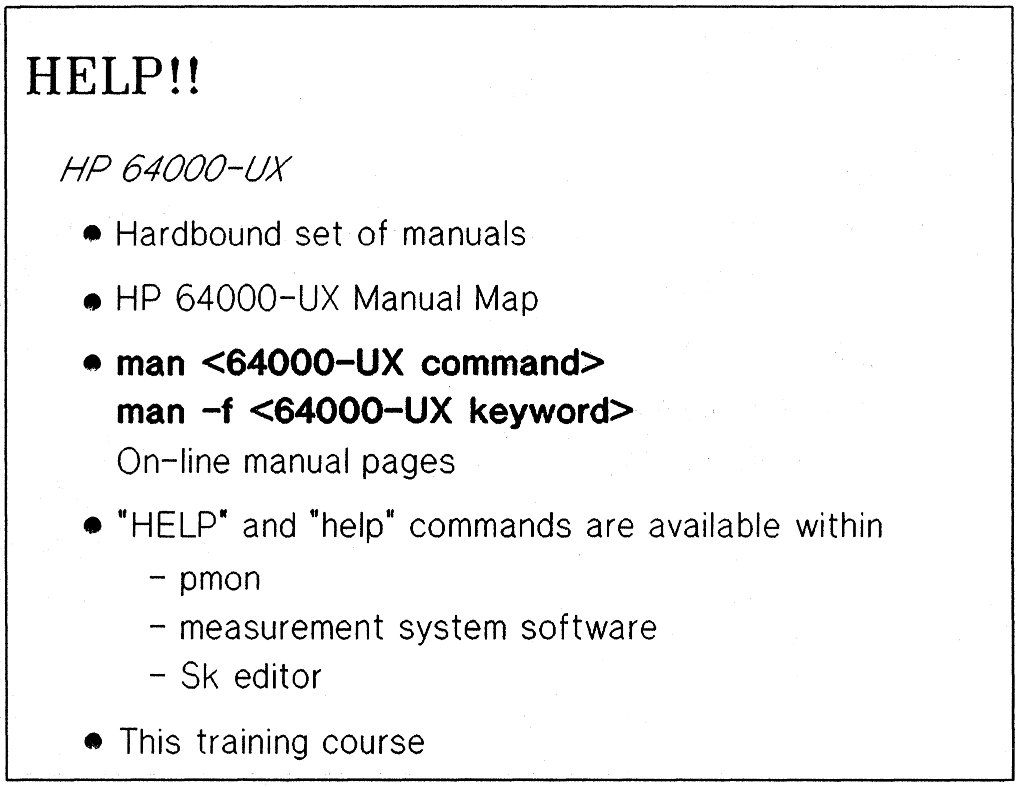
The HP-UX system manuals which contain the HP-UX command references are listed below:

HP-UX Reference
Sections 1M, 2, 3, 4, 5, and 7
Sections 1 and 9

Both manuals are listed as a single part number (HP P/N 09000-90009).

STUDENT NOTES:

□ HELP!!



HELP!!

HP 64000-UX

- Hardbound set of manuals
- HP 64000-UX Manual Map
- **man <64000-UX command>**
  man -f <64000-UX keyword>
  On-line manual pages
- "HELP" and "help" commands are available within
  - pmon
  - measurement system software
  - Sk editor
- This training course

OV1 4.20 © 1987 Hewlett-Packard Company

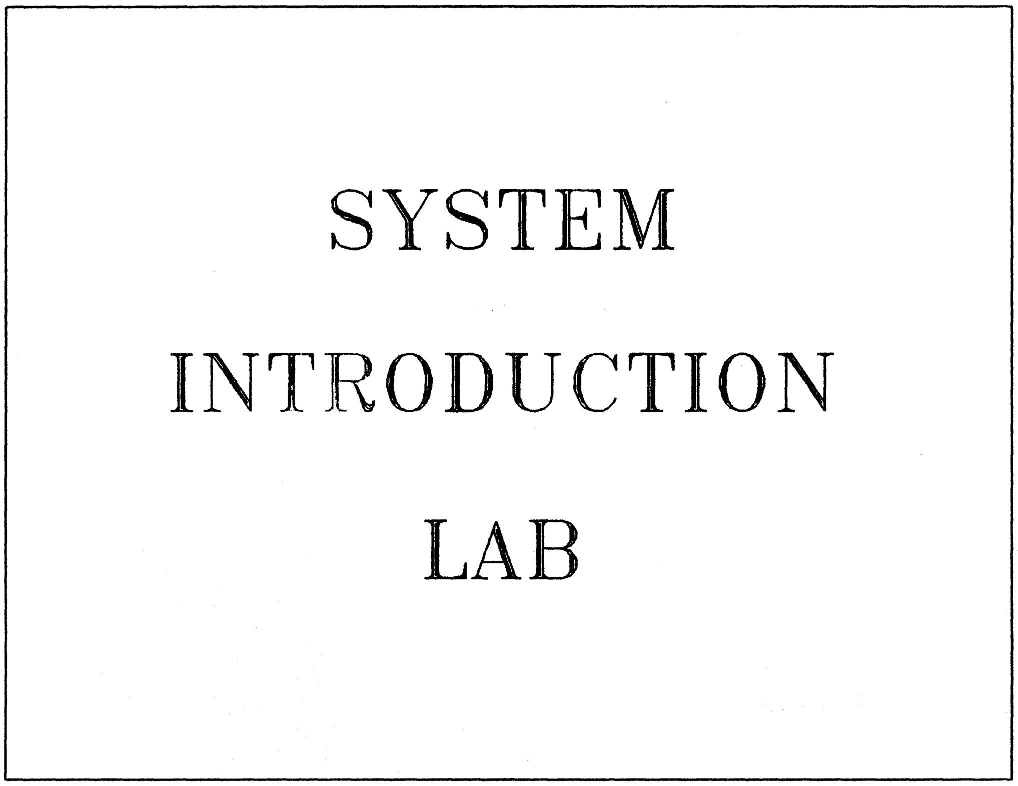
NOTES:

The manuals which contain the detailed information about the HP 64000-UX commands are listed below:

User's Guide for the HP 64000-UX Microprocessor Development Environment
HP P/N 64801-90903
User Interface Software for HP 64000-UX
HP P/N 64808-90901
Softkey Driven Editor
HP P/N 64790-90901
68000/68008 Emulation with Internal Analysis for HP 64000-UX
HP P/N 64243-90903
Measurement System for HP 64000-UX Microprocessor Development Environment
HP P/N 64801-90908
Emulation/Analysis Reference Manual for 8 and 16 Bit Emulators
HP P/N 64200-90912
68000/08/10 Cross Assembler/Linker
HP P/N 64845-90905
68000/08/10 C Cross Compiler
HP P/N 64819-90903

STUDENT NOTES:

☐ System Introduction Lab



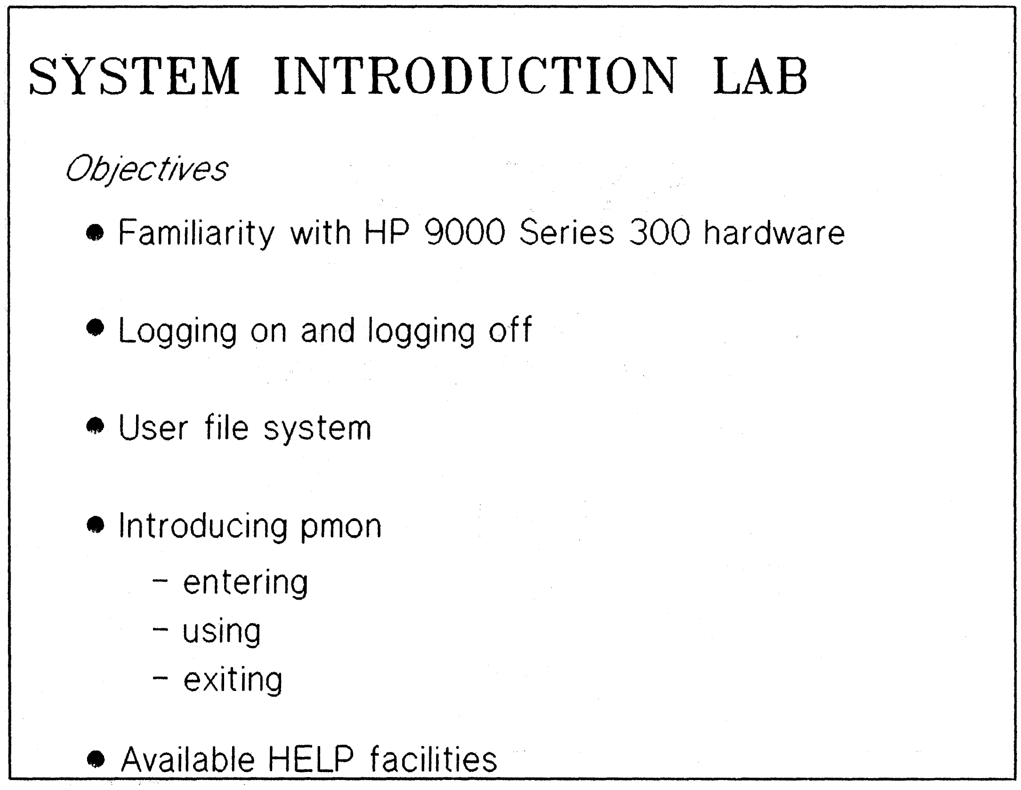
SYSTEM

INTRODUCTION

LAB

ISI 1.55

☐ System Introduction Lab



SYSTEM INTRODUCTION LAB

Objectives

- Familiarity with HP 9000 Series 300 hardware
- Logging on and logging off
- User file system
- Introducing pmon
  - entering
  - using
  - exiting
- Available HELP facilities

ISI 1.60

STUDENT NOTES:

HP 64000-UX System Introduction Lab

Objectives:

* familiarity with the 9000 series 300 hardware
* logging on and logging off
* the user file system
* pmon (user interface software)
* available help facilities

Each of the sections of the lab are supplied with a box. This box should be checked by the student when the tasks or suggestions of each of these sections are completed.

HP 9000 Series 300 Computer

[ ] Survey the computer hardware of the 9000 series 300 computer. Locate the:

- computer monitor
- computer keyboard
- mouse
- computer CPU box
- system disc(s)
- LAN cable (local area network)
- RS-232 9-pin modem port
- HP-IB interface cables for:
  - disc
  - printer
  - 64000-UX instrumentation cardcage

Notes:
The LAN cable is a small coaxial cable connected to the computer through a "T" type connection.

The HP-IB interfaces are for peripheral connections to the computer.
A high speed HP-IB interface is used to connect the system disc.
A slow speed HP-IB interface is used to connect a system line printer.
A third HP-IB interface is used as a dedicated connection to the 64000-UX cardcage (64120).

Logging On

[ ] Log on the 9000 using the login name and password supplied by the instructor.

☐ System Introduction Lab

User Interface Software

[ ] Type the command "pmon" to start the user interface software. You will notice two portions of the screen, the softkeys and the status line. The status line begins by showing the revision of software you have on the system.

Push the "---ETC--" softkey. You will notice that the softkey labels have changed and the status line now shows your current working directory. Push the "---ETC--" softkey two more times and you see the same keys once again.

[ ] Exit pmon by using one of the methods mentioned below.
There are two ways to exit pmon:

1. type the command "end" on the command line
   NOTE: although a softkey does not exist for "end" it can be typed in on the command line.
2. use the end-of-file control sequence "cntrl d".

Help Facilities

[ ] Enter pmon again.

Help is readily available inside pmon for both the 64000-UX commands and HP-UX commands.

[ ] Press the "HELP" softkey and determine the method for gaining on-line help with HP-UX.

[ ] Pmon is so friendly, it even offers help with syntax. Let's check the system date with the help of pmon. When the "date&time" softkey is pushed, only one new softkey appears; "<DATE>". Anytime a softkey appears with greater than and less than signs (< >) surrounding it, the user is expected to provide input to the command whether it be a filename or the date and time.

[ ] Press the "<DATE>" key and look at the information on the status line. Pmon is smart enough to prompt you for the proper syntax of user input! Now press the return key to see the current date and time.

[ ] One last help facility is the "help" command. Type the command "help" on the command line and see what happens! You now have even another way of getting help with HP-UX and 64000-UX.

Complete this help command for both the options and see what additional information is available.

☐ System Introduction Lab

HP-UX Command Mapping

[ ] Cycle the softkeys until the "list_dir" set appears. Press the softkey and note that the softkey has become an "ls" on the pmon command line. In essence, the softkey has become an equivalent HP-UX command.

[ ] Next, obtain an options summary of the "list_dir" command using the available softkeys.

[ ] Display the files of the current directory. Sort their names by modification times, the oldest file first.

As options of original command are used, the softkeys continue to change to give the user the only options to the command. This is called directed syntax.

[ ] Spend five to ten minutes experimenting with the various levels of softkeys for the HP-UX commands.

NOTE: The following softkeys are used as part of the HP 64000-UX development system and will be discussed later:

```
opt_test
MEAS_SYS
prom_prg
compile
assemble
link
```

NOTE: The following softkeys have more complicated functions and will be discussed later in the course:

```
edit
log
shell
-BACKUP-
```

☐ System Introduction Lab

Pmon Special Features

[ ] The user interface software has a very helpful feature which could prove to be very timesaving to an experienced user; the idea of command completion. What this allows the user to do is type the first few characters of a pmon command and touch the "tab" key. Pmon will automatically complete the full command name for you!

As an example type these letters on the command line: "lo" and press the tab key. You will notice that the command for "log_commands" has appeared. Now compare this to pressing the softkey which represents this command. This could save some time if you did not have to cycle through the softkeys looking for a specific command.

[ ] Pmon has another neat feature called command recall. You can use the control sequences "cntrl r" and "cntrl b" to cycle backward and forward through the previous commands. As an exercise, cycle back through the commands and re-execute the command for displaying the files of the current directory sorted on access time with the oldest file appearing first.

Entering HP-UX Commands

[ ] Since pmon is merely a program running on a shell, it is possible to start another process and temporarily leave the software. Use the help facilities to determine how to start this temporary process and then check and see what your current working directory is. HINT: "pwd" command.

NOTE: when changing directories within pmon, never issued the "cd" command preceded by a "!". This will create a new shell change the directory for the new shell and will return to the parent shell. In essence, the directory is changed, but not in the shell the user desired.

☐ System Introduction Lab

[ ] Next, use this new feature to TEMPORARILY exit pmon by creating a shell. Hint: look at the pmon "shell" command.

Try an HP-UX "ls" command. Notice this was done under the HP-UX operating system and not under pmon.

Once you have successfully started another shell out of pmon, kill that shell with a "cntrl d" or an "exit". Where did you end up? As you can see this is a very helpful feature of pmon and can save large amounts of time not having to exit pmon, do something and then start the program over.

NOTE: if you start a shell out of pmon, DO NOT type in "pmon" to get back! This will cause an enormous number of processes to be created.

Additional HP-UX commands

[ ] If time permits, experiment with the other available HP-UX commands available within pmon listed below:

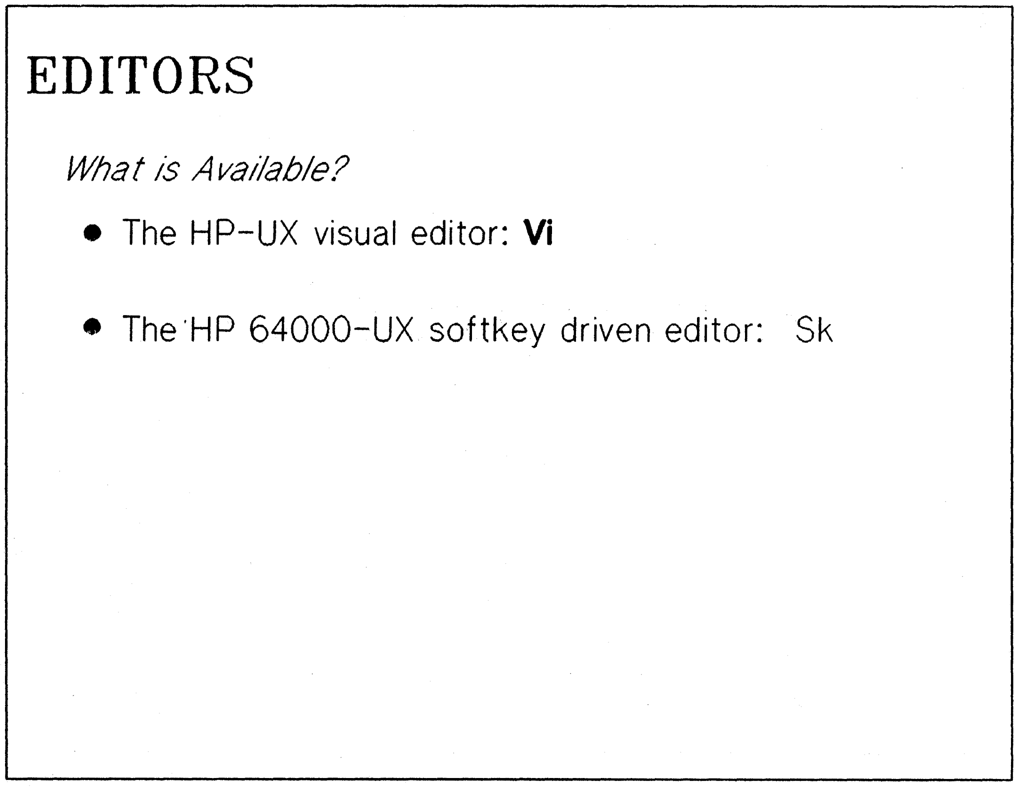
```
remove
move
copy
cat
makedir
removdir
chng_dir
date&tme
```

□ Editors

EDITORS

OV1 6.05 © 1987 Hewlett-Packard Company

☐ Editors



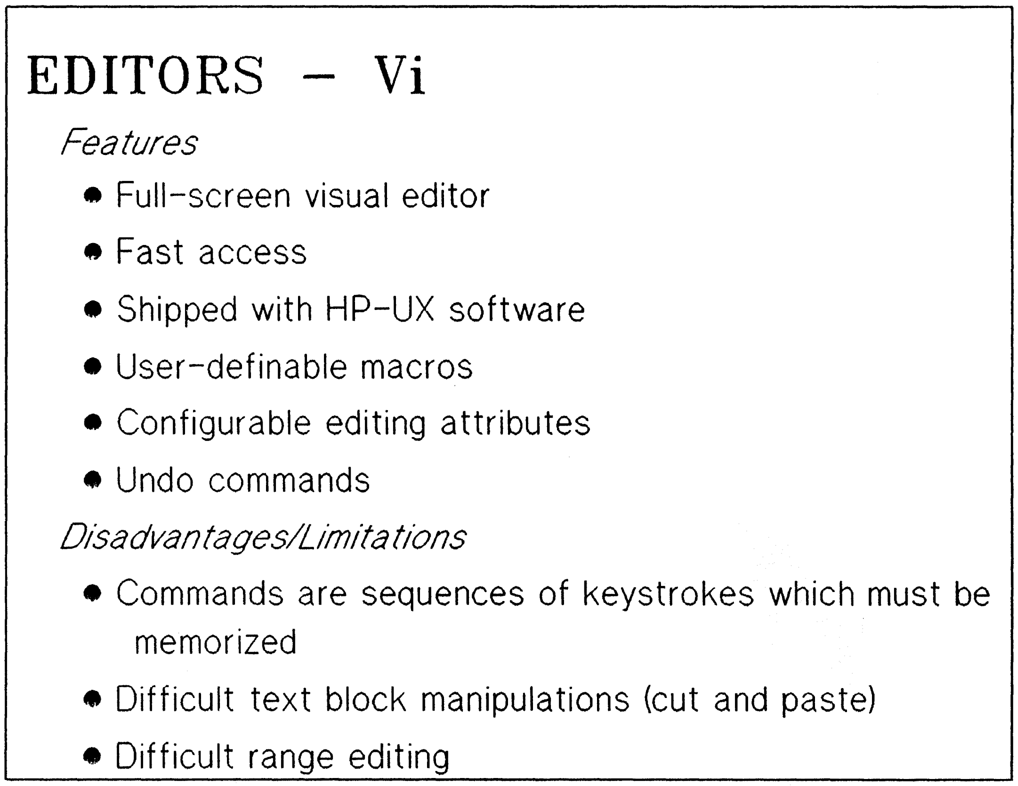
EDITORS

What is Available?

- The HP-UX visual editor: **Vi**
- The HP 64000-UX softkey driven editor: Sk

OVI 6.10

STUDENT NOTES:



EDITORS – Vi

Features

- Full-screen visual editor
- Fast access
- Shipped with HP-UX software
- User-definable macros
- Configurable editing attributes
- Undo commands

Disadvantages/Limitations

- Commands are sequences of keystrokes which must be memorized
- Difficult text block manipulations (cut and paste)
- Difficult range editing

OVI 6.15 © 1987 Hewlett-Packard Company

NOTES:

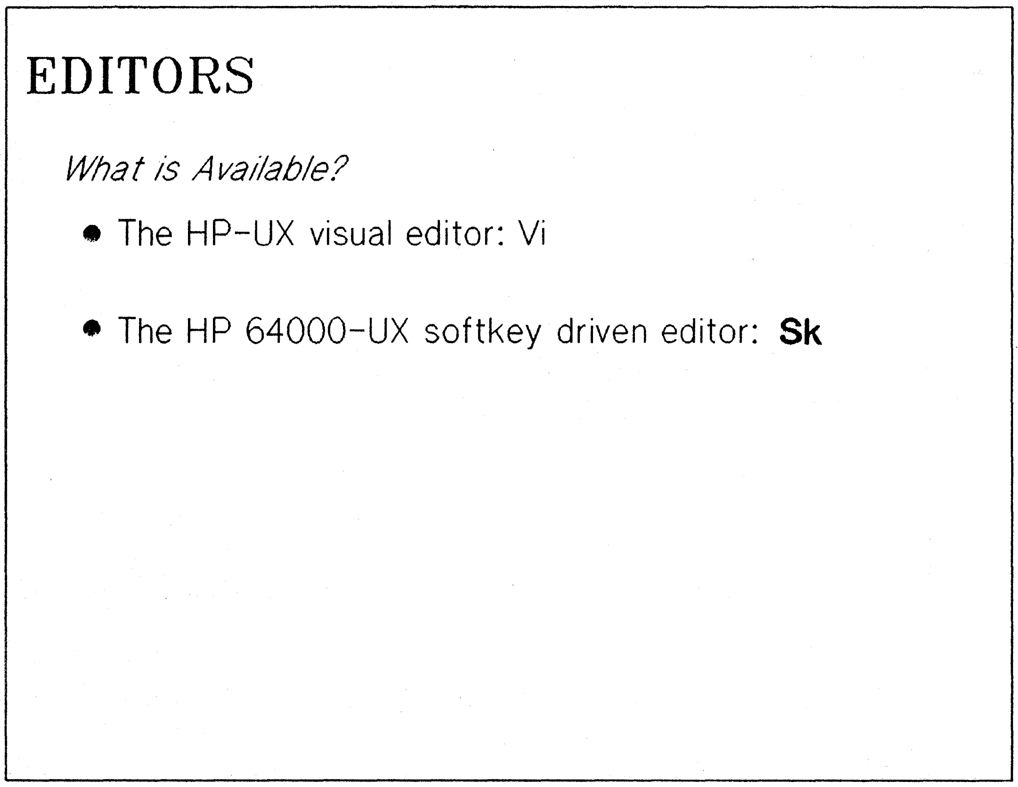
The biggest drawbacks associated with the Vi editor are the cryptic commands which must be memorized to become proficient with the editor. There is a considerable amount of time required to learn Vi.

One of the nicest features of the Vi editor is the "undo" command. This allows the user to undo the last insertion of text or deletion of text.

More information is available on Vi in the following manual:

Text Editors and Processors
HP-UX Concepts and Tutorials
HP P/N 97089-90022

STUDENT NOTES:



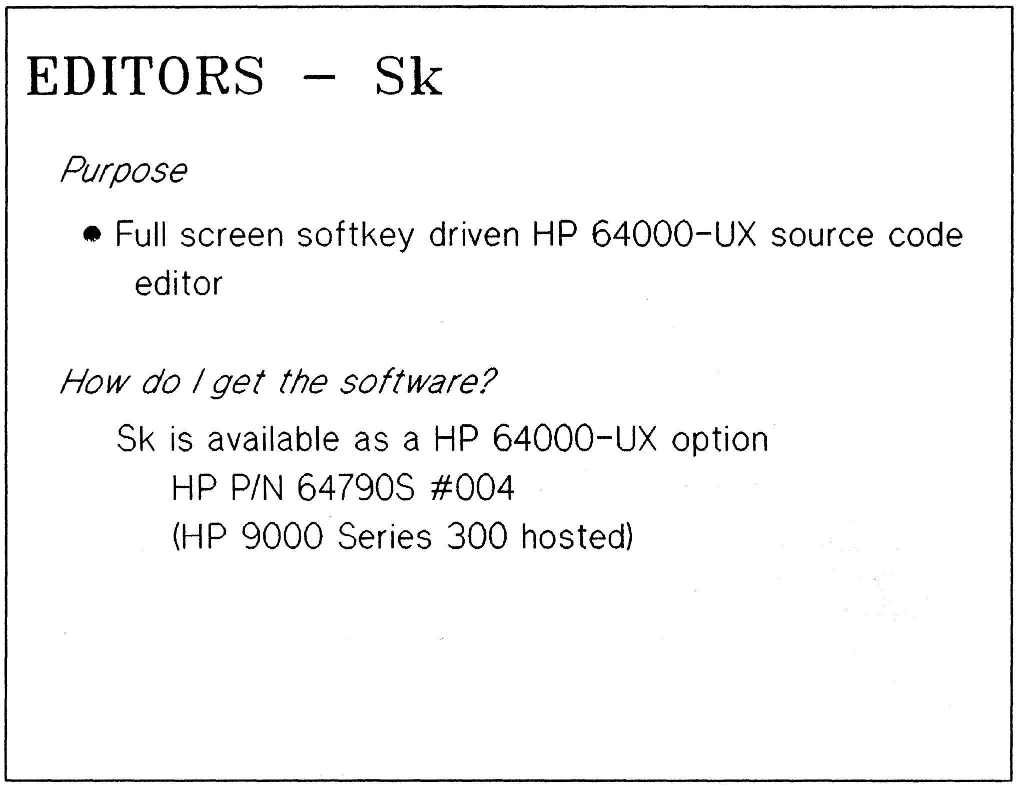
EDITORS

What is Available?

- The HP-UX visual editor: Vi
- The HP 64000-UX softkey driven editor: **Sk**

OVI 6.20

STUDENT NOTES:



EDITORS – Sk

Purpose

- Full screen softkey driven HP 64000-UX source code editor

How do I get the software?

Sk is available as a HP 64000-UX option
HP P/N 64790S #004
(HP 9000 Series 300 hosted)

OVI 6.25 ○ 1987 Hewlett-Packard Company

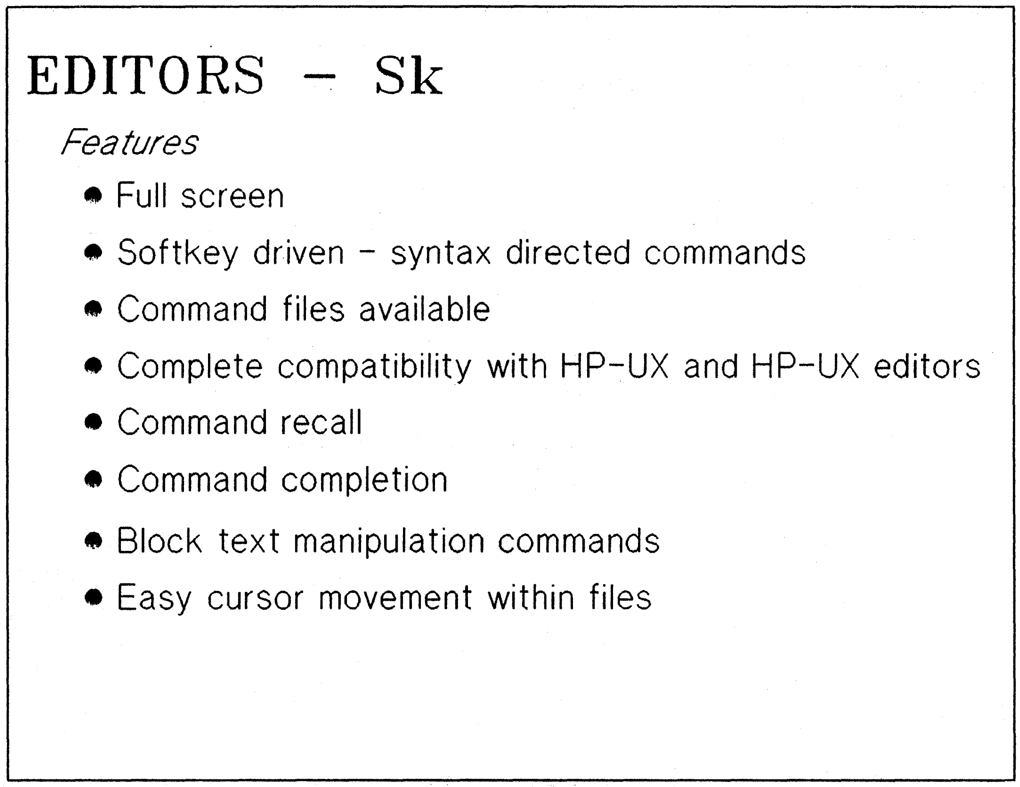
MANUAL: page 3-1 Softkey Driven Editor

NOTES:

The Sk editor is available as an option to the 64000-UX Development Environment. Contact your local Hewlett-Packard sales representative (FE) for more information.

STUDENT NOTES:

☐ Editors



EDITORS – Sk

Features

- Full screen
- Softkey driven – syntax directed commands
- Command files available
- Complete compatibility with HP-UX and HP-UX editors
- Command recall
- Command completion
- Block text manipulation commands
- Easy cursor movement within files

OVI 6.30

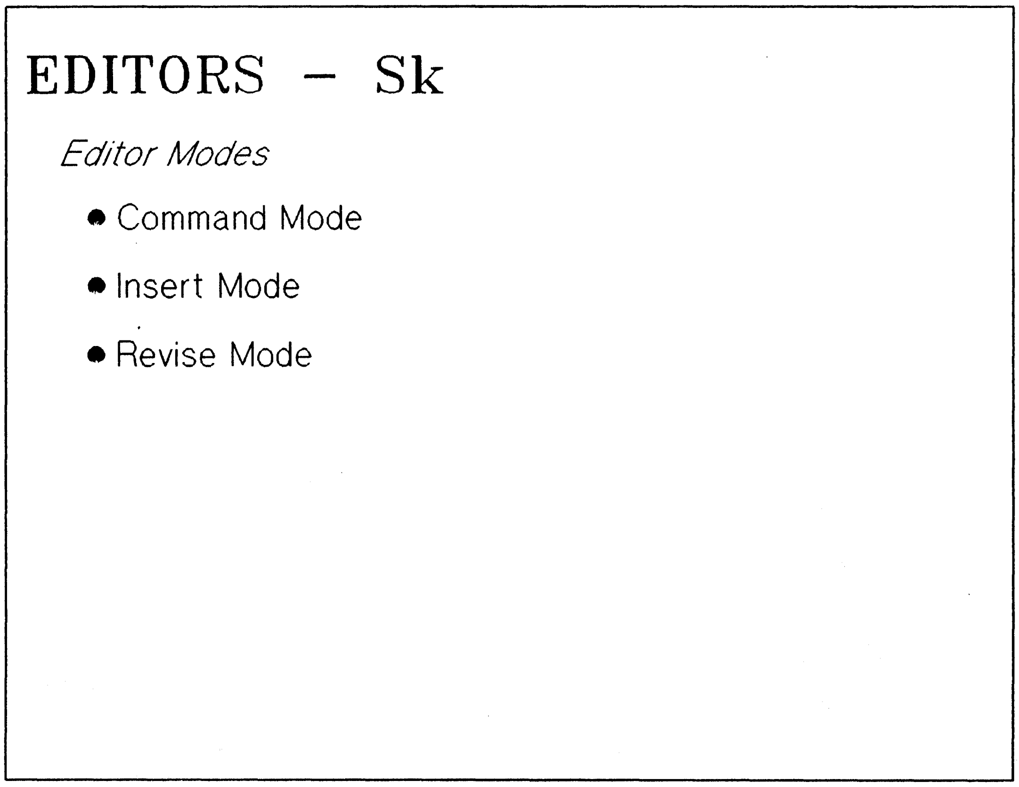
MANUAL: page 3-1 Softkey Driven Editor

NOTES:

Sk has all of the features of a conventional editor plus the different features listed below:

command recall
command completion
softkey driven - syntax directed commands
block text manipulation commands
line number referencing
command files available

STUDENT NOTES:



EDITORS - Sk

Editor Modes

- Command Mode
- Insert Mode
- Revise Mode

OV1 6.35 © 1987 Hewlett-Packard Company

MANUAL: pages 3-3 thru 3-5 Softkey Driven Editor

NOTES:

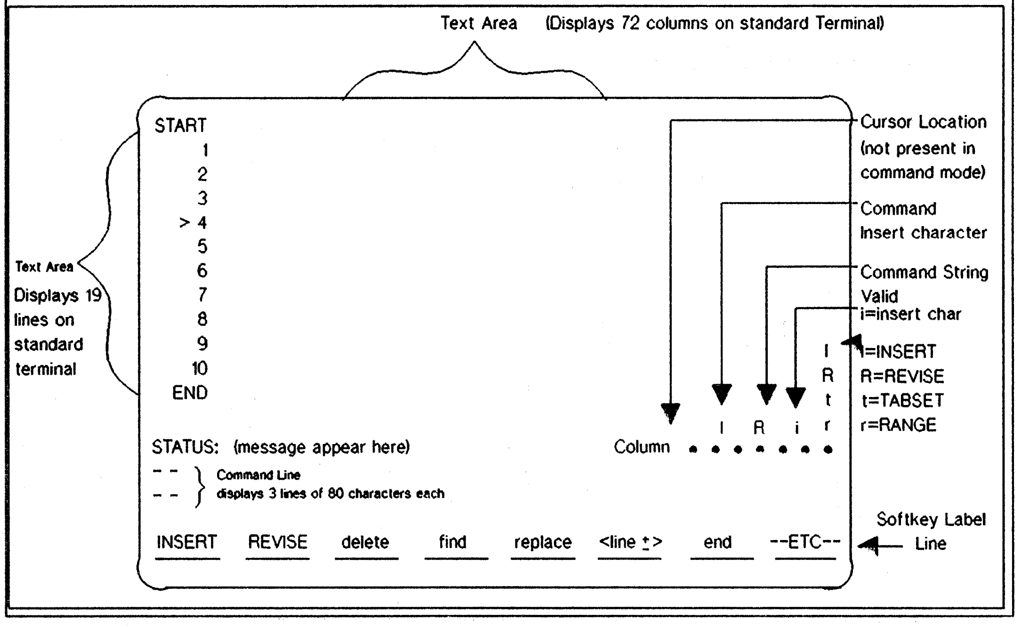
The default editor mode is "insert mode" for a new file and the "command mode" for already existing file.

STUDENT NOTES:

☐ Editors

EDITORS – Sk

What will the screen look like?


Text Area (Displays 72 columns on standard Terminal)
START
1
2
3
> 4
5
6
7
8
9
10
END
Text Area
Displays 19 lines on standard terminal
Cursor Location (not present in command mode)
Command Insert character
Command String Valid
i=insert char
I=INSERT
R=REVISE
t=TABSET
r=RANGE
I R t r
I R i r
STATUS: (message appear here)
Column
Command Line displays 3 lines of 80 characters each
INSERT
REVISE
delete
find
replace
<line ±>
end
--ETC--
Softkey Label Line

OV1 6.40 © 1987 Hewlett-Packard Company

MANUAL: pages 4-1 thru 4-6 Softkey Driven Editor

NOTES:

The Sk editor display screen is made up of the following parts:

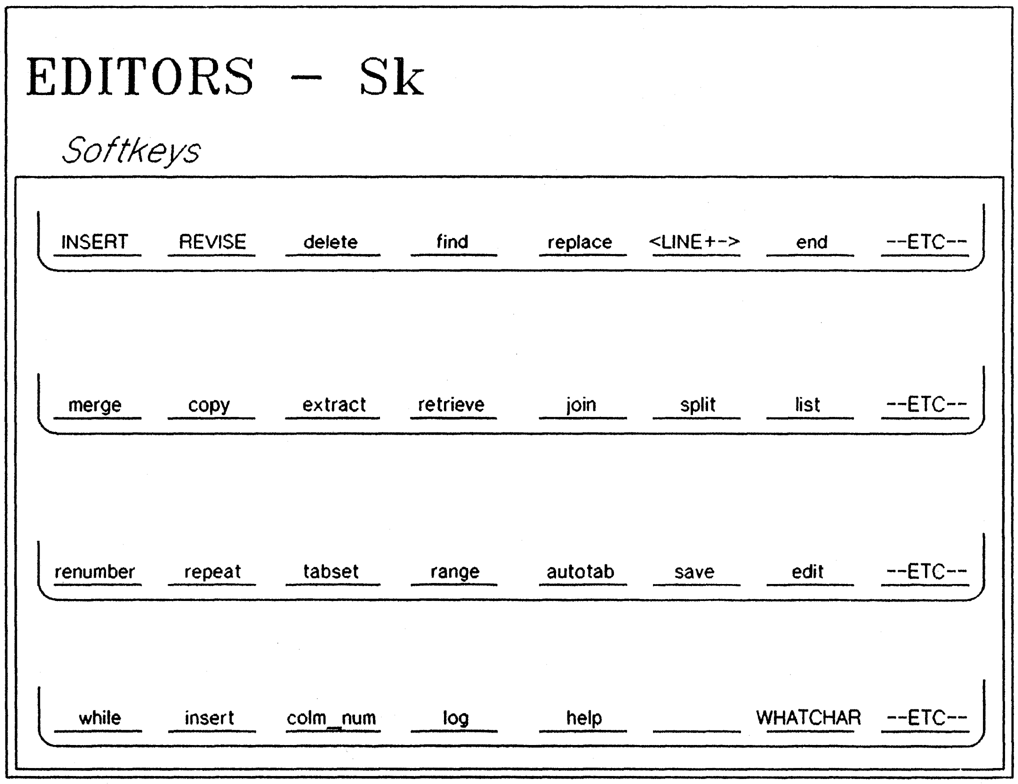
text area
line numbers
status line
command lines (3)
softkey command labels
column number and editor status indicators

The editor cursor is used to show the position of the current edit command; whether it is on the command line or among the text in the text area.

All the information given in this section will be based on a standard Hewlett-Packard terminal, such as the HP 2392A.

STUDENT NOTES:

□ Editors



EDITORS – Sk

Softkeys

INSERT	REVISE	delete	find	replace	<LINE+->	end	--ETC--
merge	copy	extract	retrieve	join	split	list	--ETC--
renumber	repeat	tabset	range	autotab	save	edit	--ETC--
while	insert	colm_num	log	help		WHATCHAR	--ETC--

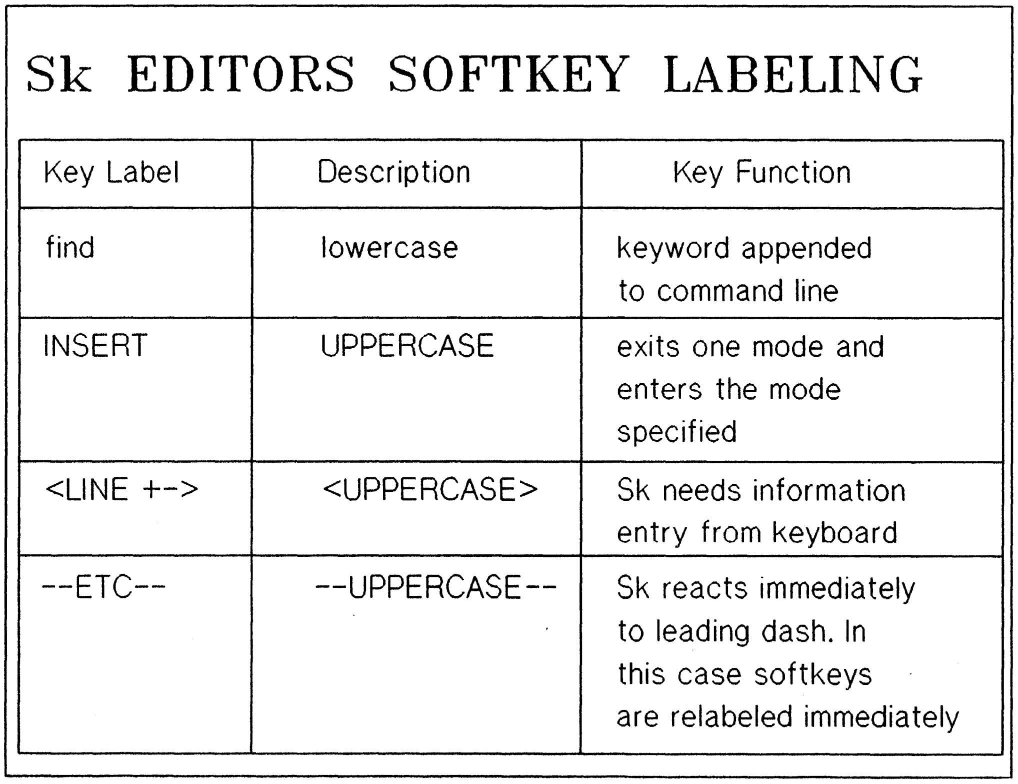
OVI 6.45

MANUAL: page 3-2 Softkey Driven Editor

NOTES:

The above diagram shows the "first" level of softkeys available for the Sk editor. In essence, there is one first level of softkeys which contains four sets.

STUDENT NOTES:



Sk EDITORS SOFTKEY LABELING

Key Label	Description	Key Function
find	lowercase	keyword appended to command line
INSERT	UPPERCASE	exits one mode and enters the mode specified
<LINE +->	<UPPERCASE>	Sk needs information entry from keyboard
--ETC--	--UPPERCASE--	Sk reacts immediately to leading dash. In this case softkeys are relabeled immediately

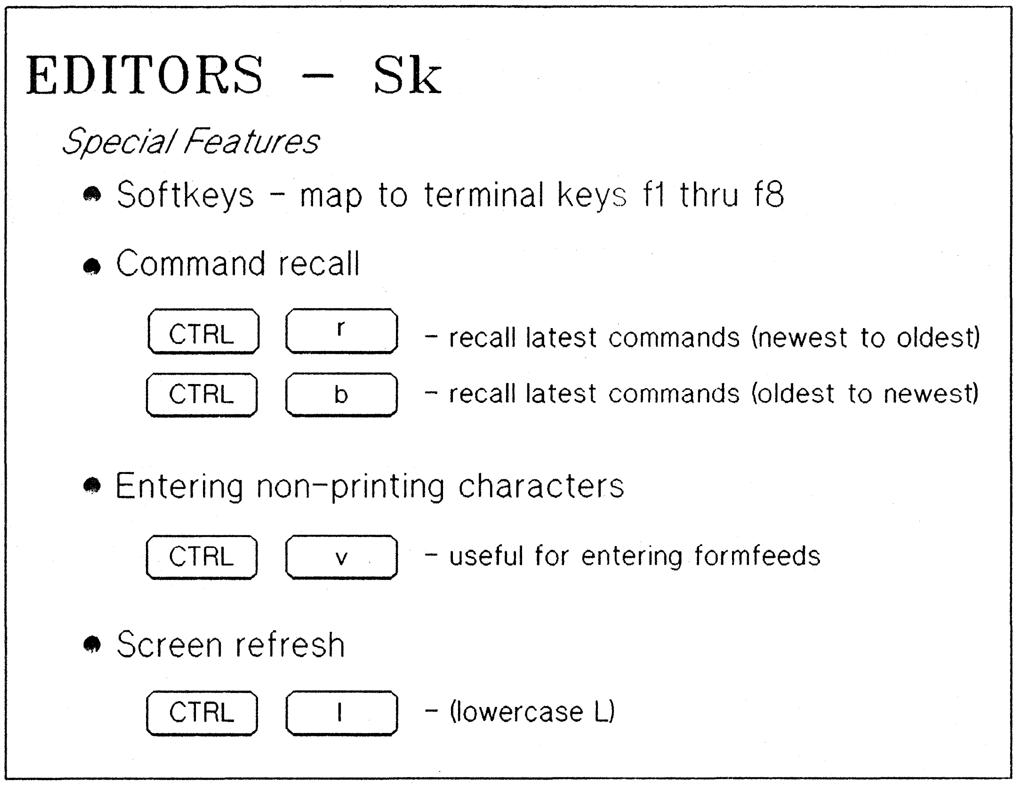
OVI 6.50

NOTES:

The table above defines the syntax of the various types of commands and softkeys of the Sk editor.

STUDENT NOTES:

□ Editors



EDITORS – Sk

Special Features

- Softkeys – map to terminal keys f1 thru f8
- Command recall

  CTRL r – recall latest commands (newest to oldest)

  CTRL b – recall latest commands (oldest to newest)
- Entering non-printing characters

  CTRL v – useful for entering formfeeds
- Screen refresh

  CTRL l – (lowercase L)

OVI 6.55 ○ 1987 Hewlett-Packard Company

MANUAL: pages 4-3 thru 4-4 Softkey Driven Editor

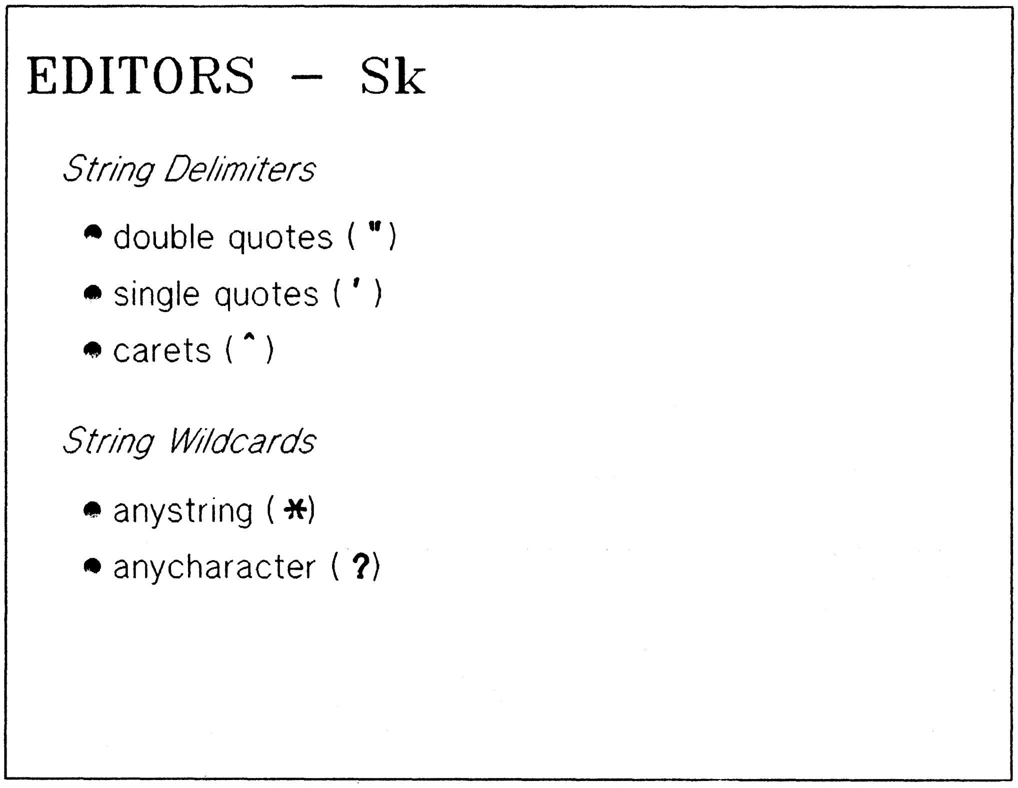
NOTES:

One of the most productive features of the Sk editor is the ability to recall commands from the command buffer.

The entering of non-printing characters becomes extremely useful when control sequences or escape sequences have to be used in a file, i.e. "formfeeds" can be entered before sending a document to a printer.

STUDENT NOTES:

☐ Editors



EDITORS – Sk

String Delimiters

- double quotes (")
- single quotes (')
- carets (^)

String Wildcards

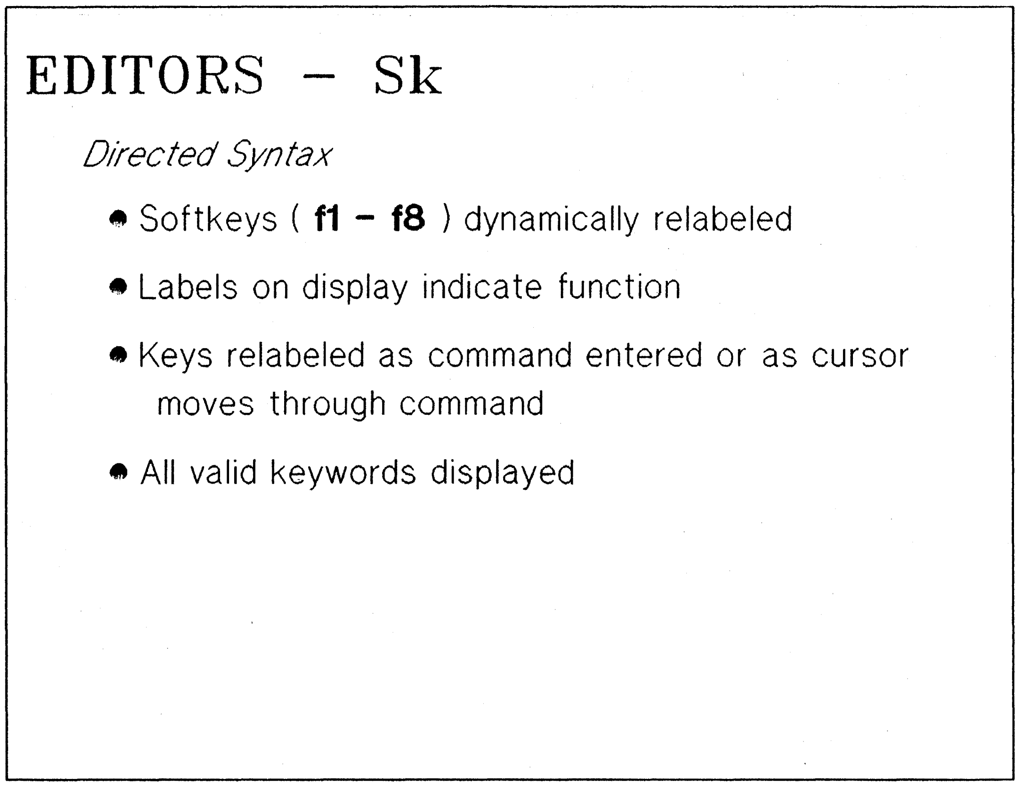
- anystring (*)
- anycharacter (?)

OV1 6.60 ○ 1987 Hewlett-Packard Company

NOTES:

The string delimiters and string wildcards of the Sk editor shown above match the same characters used by HP-UX for these functions.

STUDENT NOTES:



EDITORS – Sk

Directed Syntax

- Softkeys (**f1 – f8**) dynamically relabeled
- Labels on display indicate function
- Keys relabeled as command entered or as cursor moves through command
- All valid keywords displayed

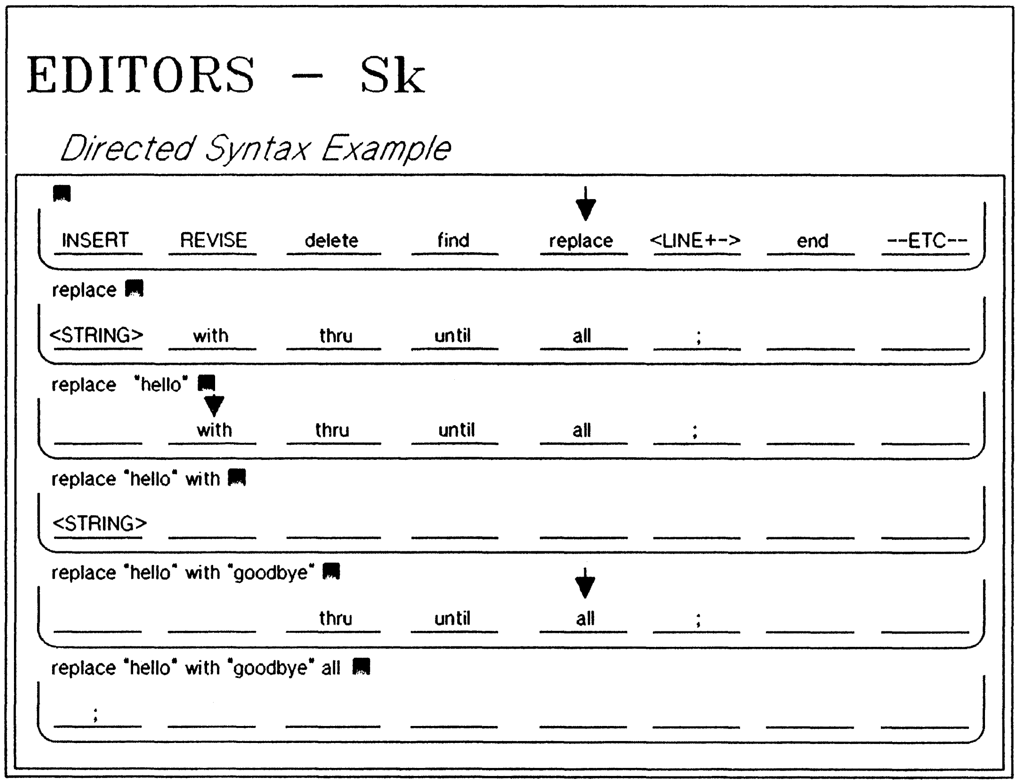
OVI 6.65 ◯ 1987 Hewlett-Packard Company

NOTES:

directed syntax: this is a feature of Sk which is used in conjunction with the softkeys to make command entry and command syntax as easy to use and remember as possible. As a command is entered, the softkeys change dynamically to show the user what syntax is appropriate. This aids in the building of complex commands and allows inexperienced users of the Sk editor to become more proficient with Sk in less time.

STUDENT NOTES:

Editors



EDITORS – Sk

Directed Syntax Example

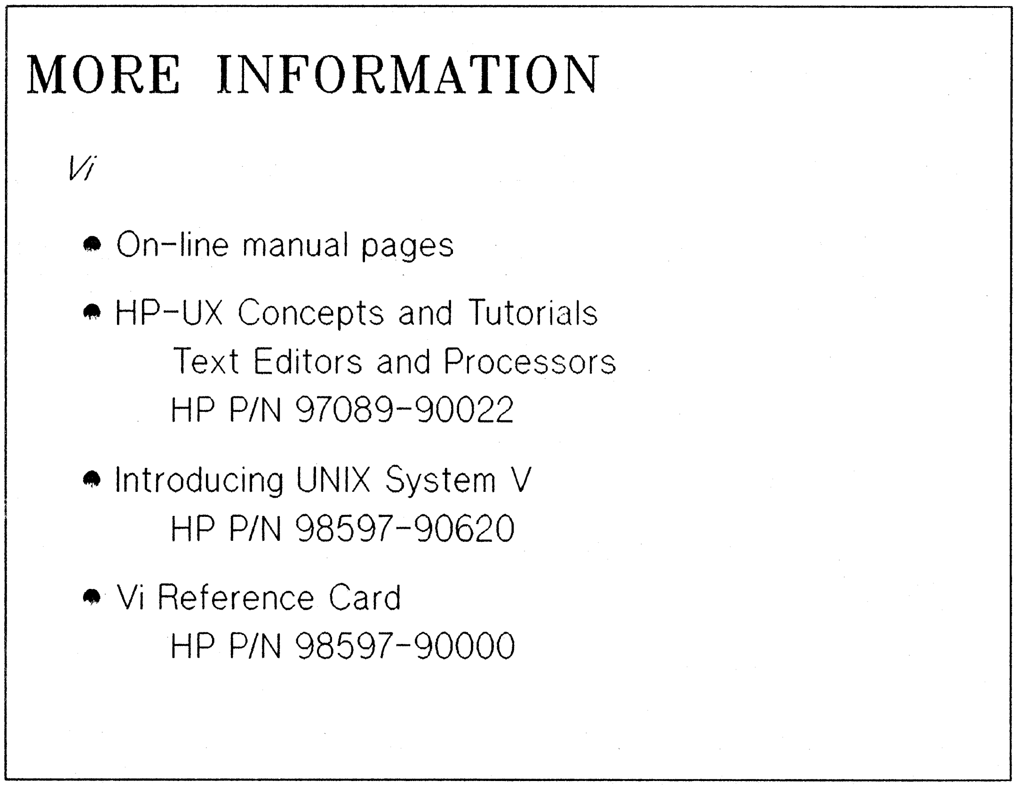
```
■
INSERT   REVISE   delete   find   replace   <LINE+->   end   --ETC--

replace ■
<STRING>   with   thru   until   all   ;

replace "hello" ■
           with   thru   until   all   ;

replace "hello" with ■
<STRING>

replace "hello" with "goodbye" ■
                  thru   until   all   ;

replace "hello" with "goodbye" all ■
   ;
```

OVI 6.70 © 1987 Hewlett-Packard Company

NOTES:

Let's take a look at an example of replacing one word with a different word throughout an entire file. This could be useful when changing the name of a symbol or an opcode in an assembly source file.

STUDENT NOTES:



MORE INFORMATION

Vi

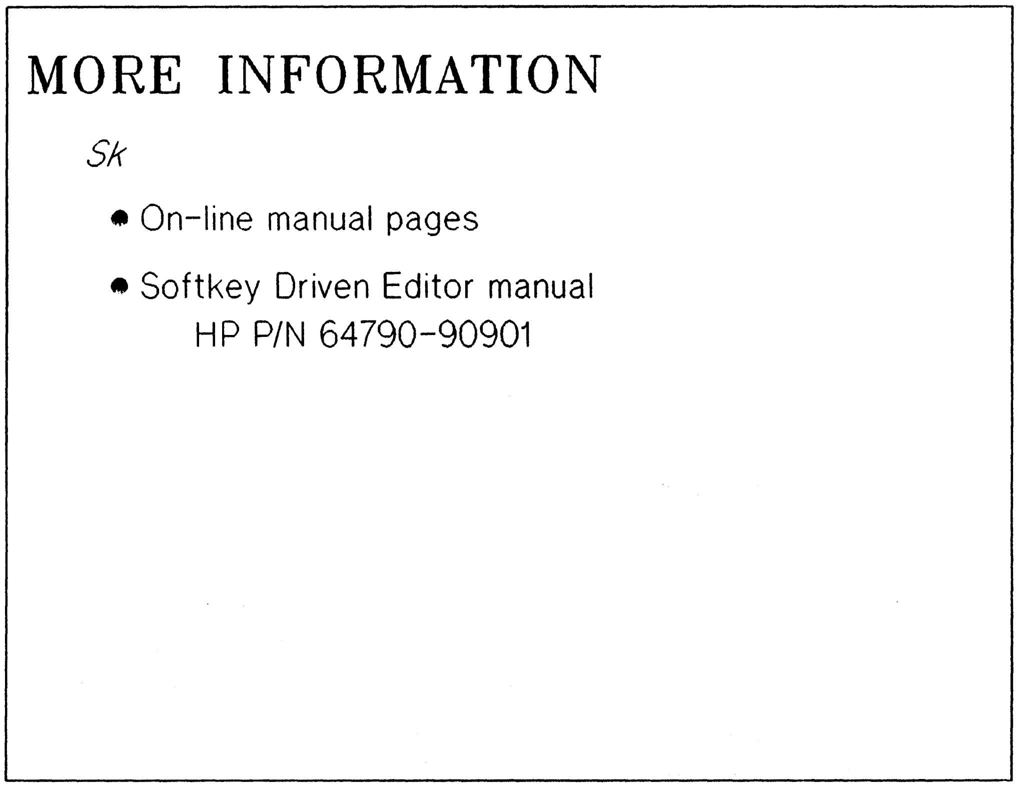
- On-line manual pages
- HP-UX Concepts and Tutorials
  Text Editors and Processors
  HP P/N 97089-90022
- Introducing UNIX System V
  HP P/N 98597-90620
- Vi Reference Card
  HP P/N 98597-90000

NOTES:

All of the resources above should be made available to you during the training course. The "on-line manual pages" and the Concepts and Tutorials manuals were also shipped with your HP 9000 series 300 computer.

STUDENT NOTES:

Editors



MORE INFORMATION

Sk

- On-line manual pages
- Softkey Driven Editor manual
  HP P/N 64790-90901

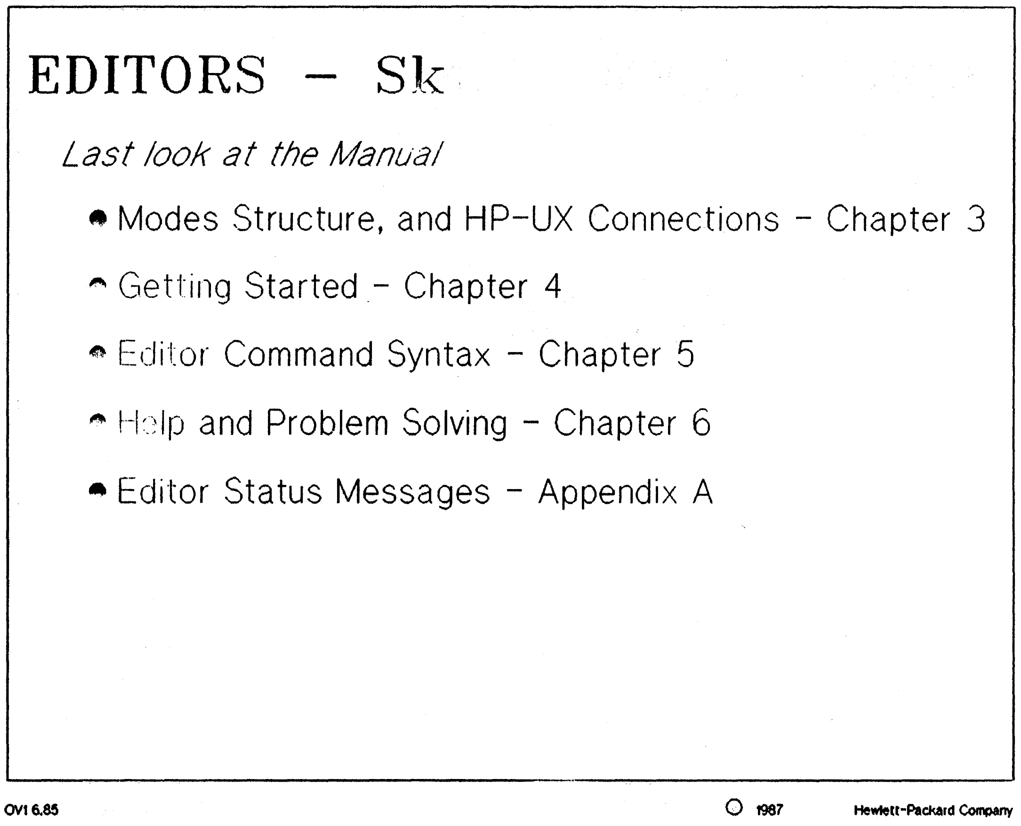
OVI 6.80

NOTES:

As is the case most of the time, your best reference for HP 64000-UX products are the manuals supplied during this course and with your initial order. Knowing where to look in the manuals is one of the more important points of being successful with a complicated system such as 64000-UX.

STUDENT NOTES:

☐ Editors



EDITORS – Sk

Last look at the Manual

- Modes Structure, and HP-UX Connections – Chapter 3
- Getting Started – Chapter 4
- Editor Command Syntax – Chapter 5
- Help and Problem Solving – Chapter 6
- Editor Status Messages – Appendix A

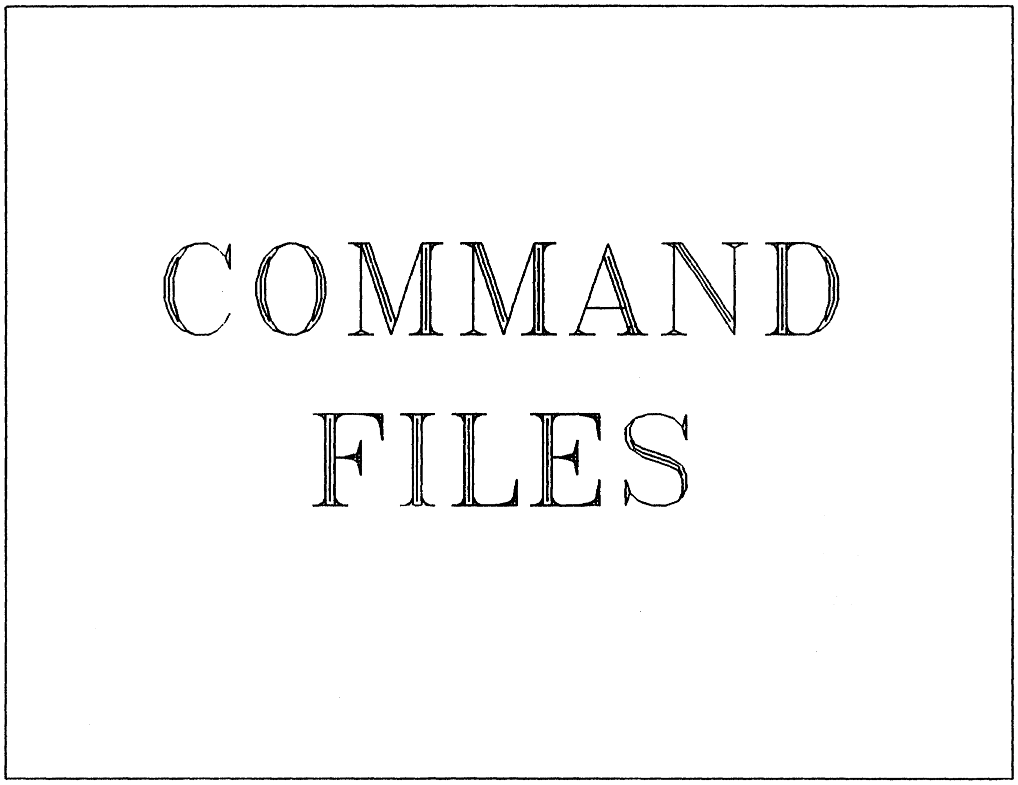
OV1 6.85 ○ 1987 Hewlett-Packard Company

NOTES:

Let's take a last look at the sections of the Sk manual.

STUDENT NOTES:

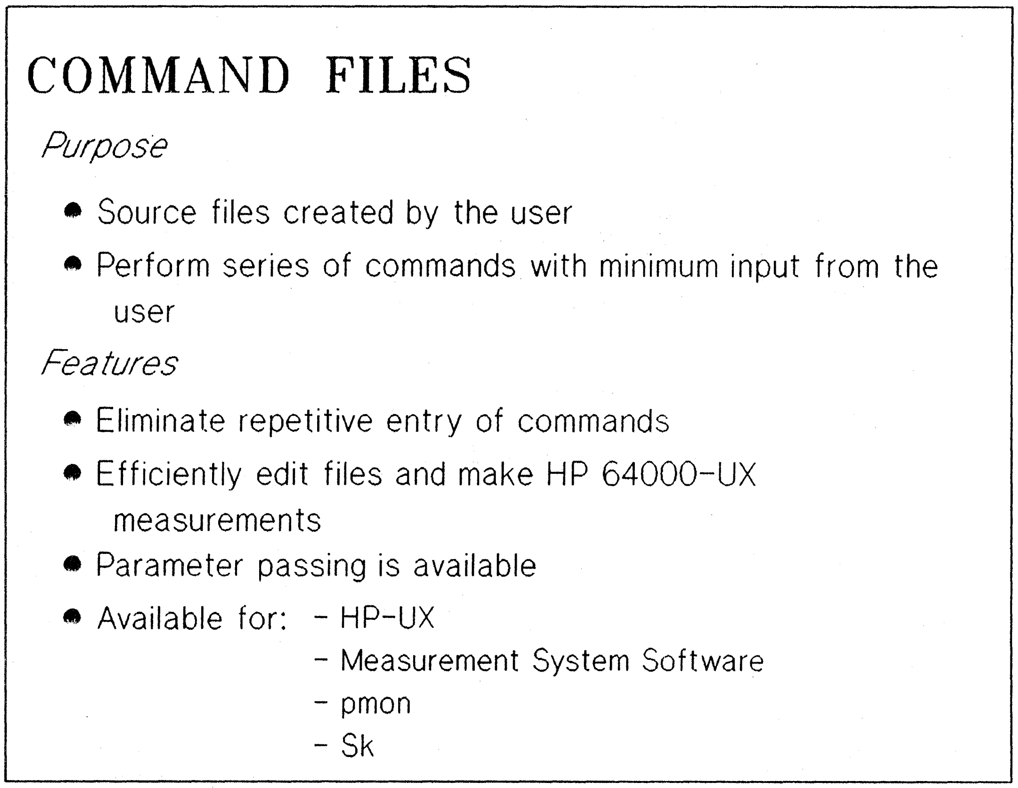
□ Command Files



COMMAND FILES

OV1 7.05

□ Command Files



COMMAND FILES

Purpose

- Source files created by the user
- Perform series of commands with minimum input from the user

Features

- Eliminate repetitive entry of commands
- Efficiently edit files and make HP 64000-UX measurements
- Parameter passing is available
- Available for: - HP-UX
  - Measurement System Software
  - pmon
  - Sk

OVI 7.10 ○ 1987 Hewlett-Packard Company

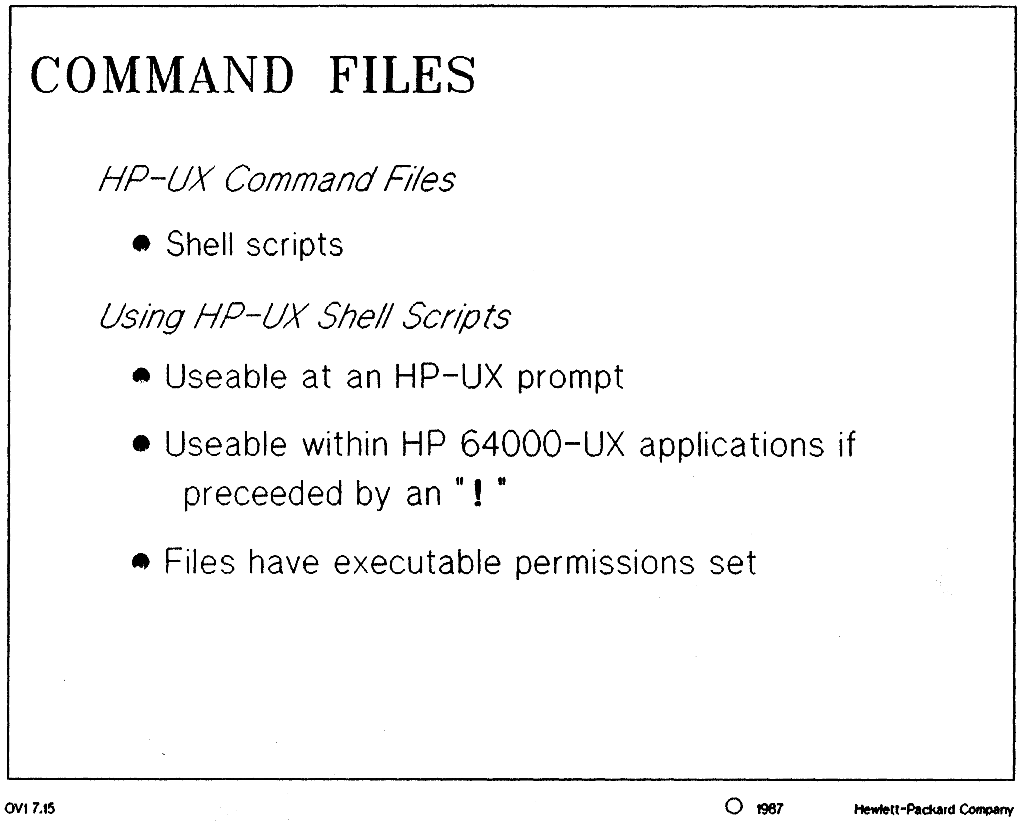
MANUAL: pages 7-1 thru 7-2 HP 64000-UX User's Guide

NOTES:

The command files for pmon, Sk, and the Measurement System software all have the same command formats. These command files can be used to call the other HP 64000-UX applications, i.e. a pmon command file can be used to call the Sk editor and perform a series of edits.

STUDENT NOTES:

☐ Command Files



COMMAND FILES

HP-UX Command Files

- Shell scripts

Using HP-UX Shell Scripts

- Useable at an HP-UX prompt
- Useable within HP 64000-UX applications if preceeded by an "**!**"
- Files have executable permissions set

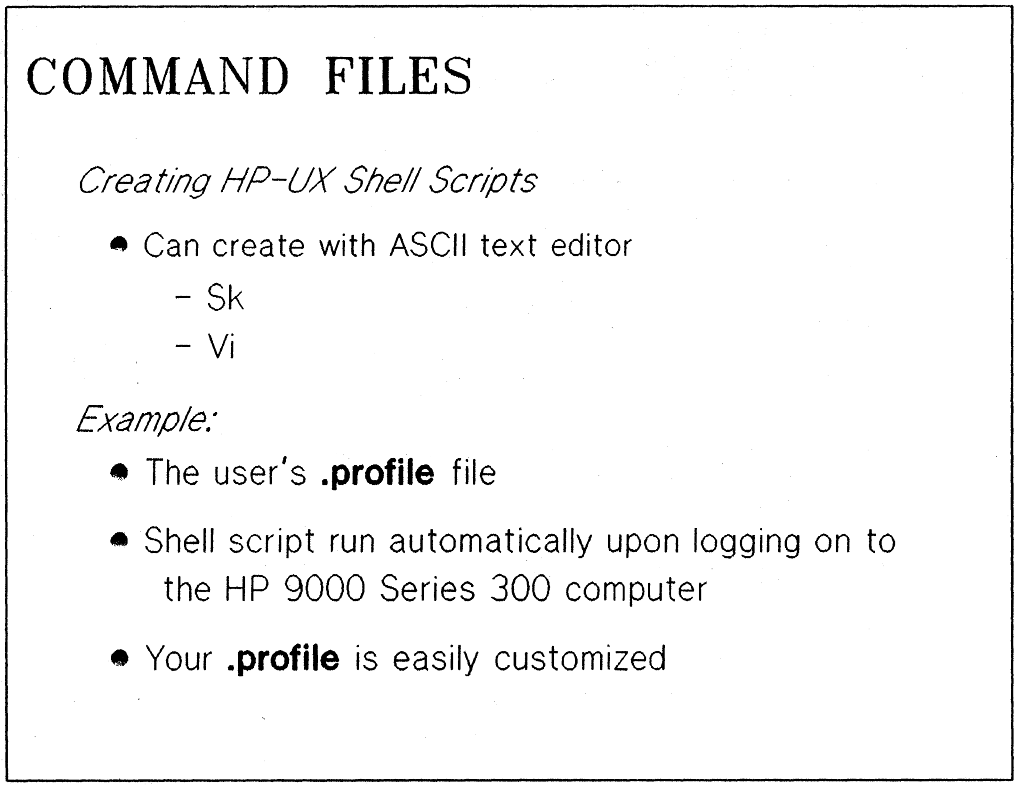
OV1 7.15

NOTES:

Shell scripts are a very powerful tool of the HP-UX Operating System. They will be mentioned in this course, but only time and experience can help a user write efficient shell scripts.

Being able to call HP-UX shell scripts from a 64000-UX application gives the user even greater flexibility using 64000-UX with HP-UX.

STUDENT NOTES:



COMMAND FILES

Creating HP-UX Shell Scripts

- Can create with ASCII text editor
  - Sk
  - Vi

Example:

- The user's **.profile** file
- Shell script run automatically upon logging on to the HP 9000 Series 300 computer
- Your **.profile** is easily customized

OVI 7.20

NOTES:

.profile file: **is a command file or shell script which customizes the user's HP-UX environment during login.**

STUDENT NOTES:

☐ Command Files

COMMAND FILES

Passing Positional Parameters

- Specifed with command name
- Syntax: **<HP-UX shell script> <$1> <$2>**

Example of file: **is_it_there**

```
echo "Searching for $1 in $2"     #echoes string with values of $1
                                  and $2 substituted

grep $1 $2                        #searches for the string $1 in file
                                  $2

echo "Done"                       #indicate when finished
```

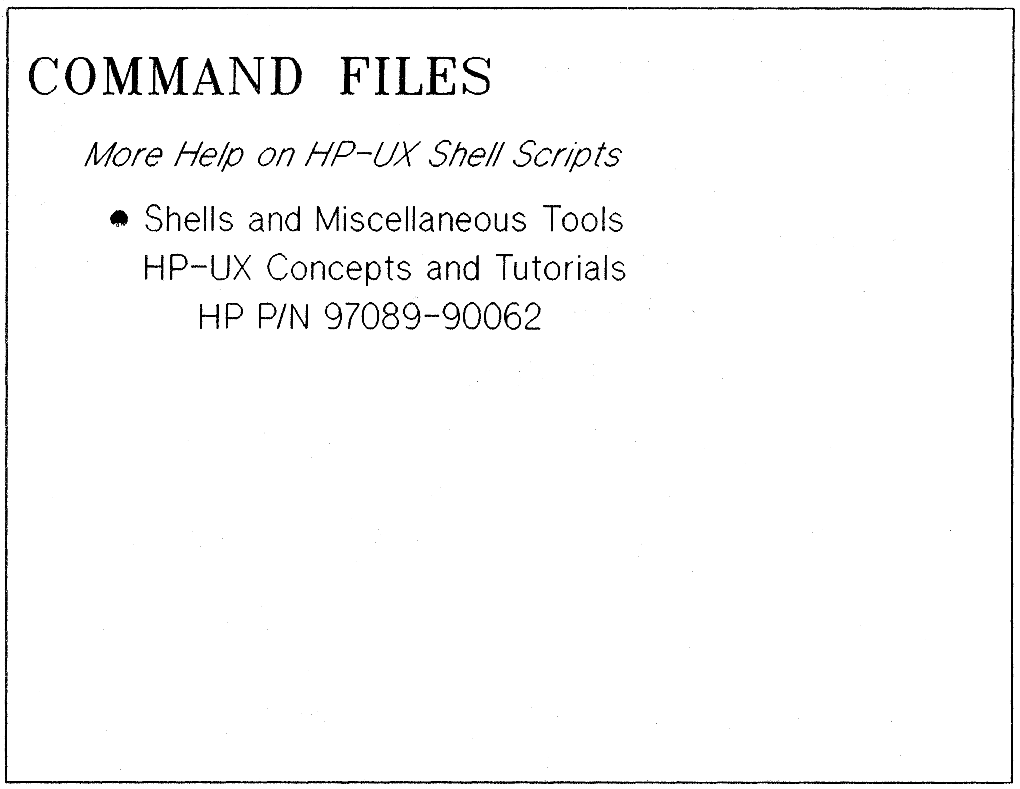
- Execute by typing: **is_it_there <$1> <$2>**

OVI 7.25 ○ 1987 Hewlett-Packard Company

NOTES:

The positional parameters of an HP-UX shell script MUST be included with the shell script command. Shell scripts cannot prompt the user for information unless they were written specifically to do that.

STUDENT NOTES:



COMMAND FILES

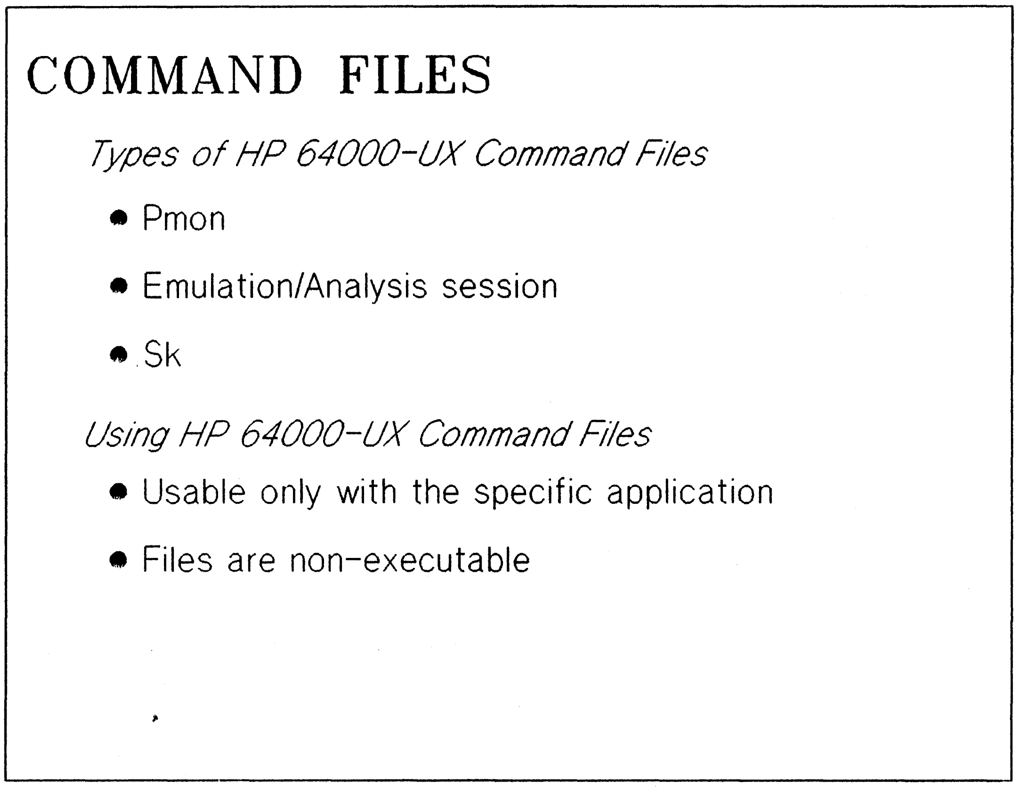
More Help on HP-UX Shell Scripts

- Shells and Miscellaneous Tools
  HP-UX Concepts and Tutorials
  HP P/N 97089-90062

OVI 7.30

STUDENT NOTES:

☐ Command Files



COMMAND FILES

Types of HP 64000-UX Command Files

- Pmon
- Emulation/Analysis session
- Sk

Using HP 64000-UX Command Files

- Usable only with the specific application
- Files are non-executable

OV1 7.35 © 1987 Hewlett-Packard Company

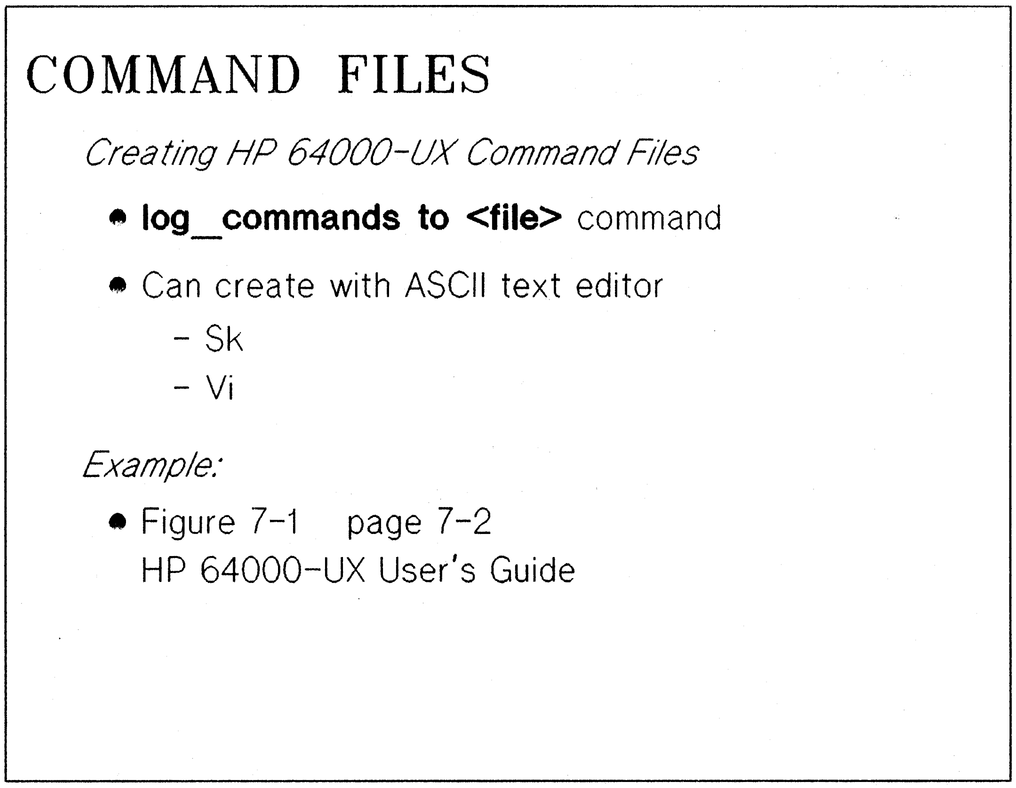
MANUAL: pages 7-1 thru 7-2 HP 64000-UX User's Guide

NOTES:

HP 64000-UX command files DO NOT have their execute permissions set and will only run properly when issued within the specific 64000-UX application.

STUDENT NOTES:

☐ Command Files



COMMAND FILES

Creating HP 64000-UX Command Files

- **log_commands to <file>** command
- Can create with ASCII text editor
  - Sk
  - Vi

Example:

- Figure 7-1 page 7-2
  HP 64000-UX User's Guide

OVI 7.40 © 1987 Hewlett-Packard Company

MANUAL: pages 7-2 thru 7-4 HP 64000-UX User's Guide

NOTES:

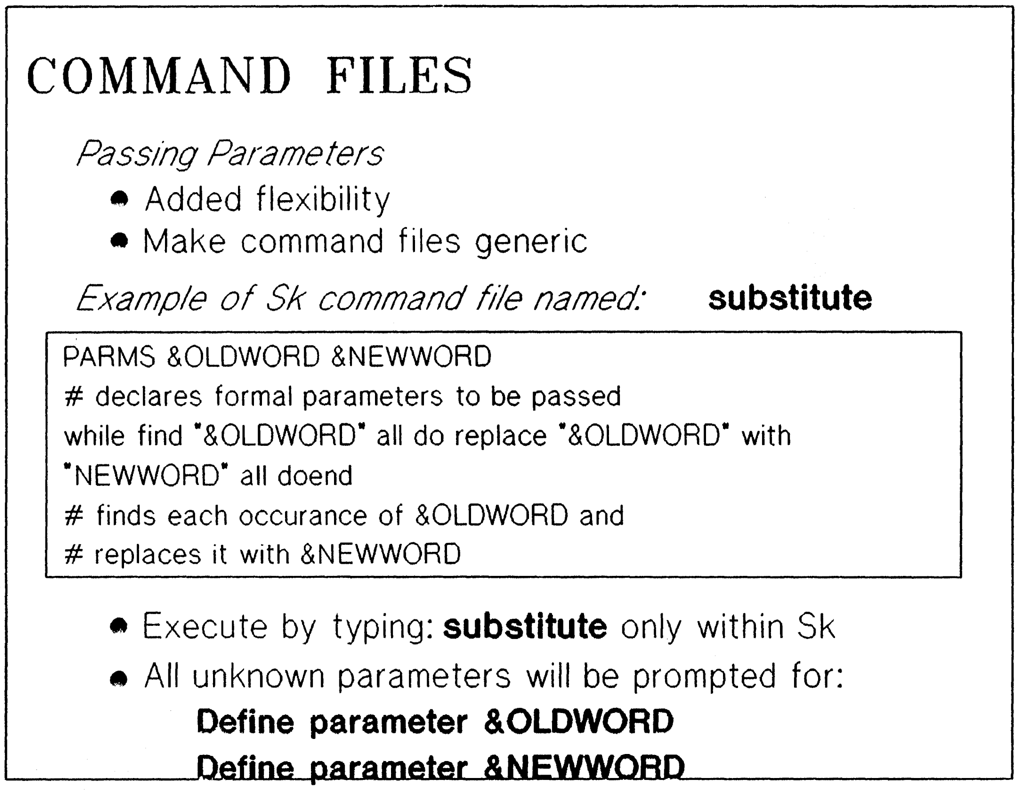
A example of a very productive 64000-UX command file is shown on page 7-2 of the User's Guide.

NOTE: the softkey label and the actual 64000-UX command usually do not have the same name. As an example, the softkey "log" translates into the 64000-UX command "log_commands". The full 64000-UX command must be used in the command file.

When using the "log_commands" command, every command issued by the user will be written into the command file, even commands which result in errors. The user is responsible for editing the commands which caused errors out of the command file.

STUDENT NOTES:

☐ Command Files



COMMAND FILES

Passing Parameters

- Added flexibility
- Make command files generic

Example of Sk command file named: **substitute**

```
PARMS &OLDWORD &NEWWORD
# declares formal parameters to be passed
while find "&OLDWORD" all do replace "&OLDWORD" with
"NEWWORD" all doend
# finds each occurance of &OLDWORD and
# replaces it with &NEWWORD
```

- Execute by typing: **substitute** only within Sk
- All unknown parameters will be prompted for:

  Define parameter &OLDWORD
  Define parameter &NEWWORD

OV1 7.45 ○ 1987 Hewlett-Packard Company

MANUAL: pages 7-5 thru 7-6 HP 64000-UX User's Guide

NOTES:

The above example will give you a good idea of the uses and power of command files.

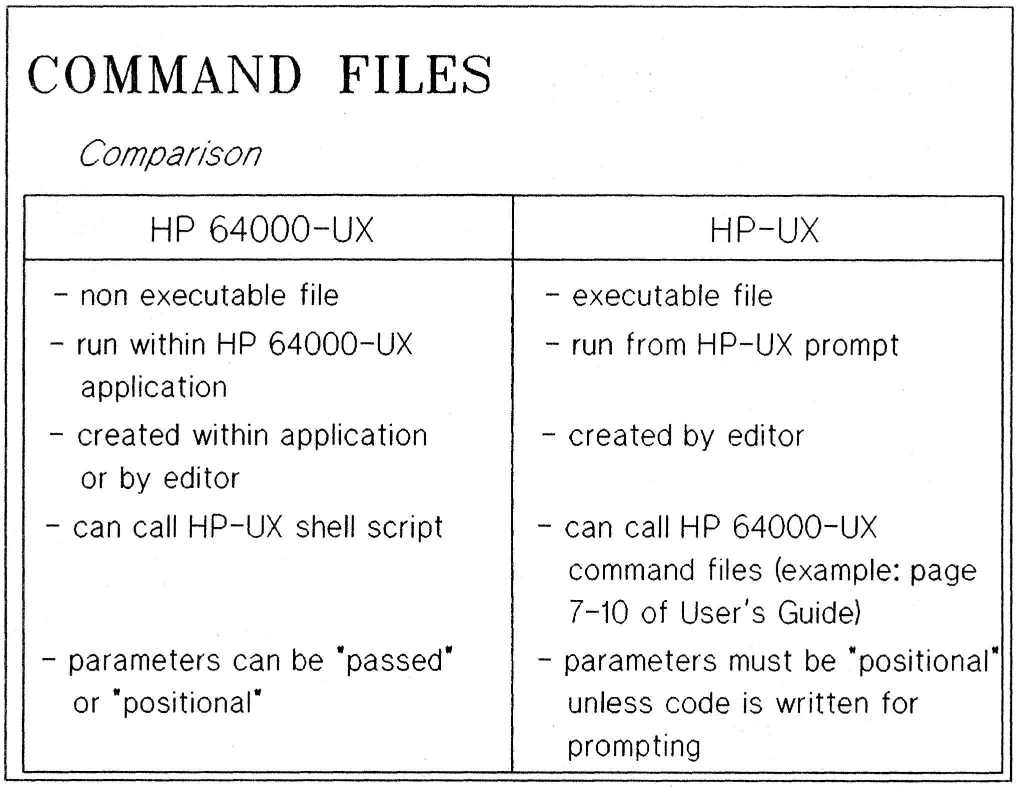
parameters: these are variables which can be changed or customized each time a command file is run.

HP 64000-UX command files will prompt the user for unknown values of parameters in the command file, unlike HP-UX shell scripts.

NOTE: the "PARMS" statement must be on the first line of the command file. It must also contain ALL the parameters associated with the command file on that one line or the command file will interpret the commands improperly.

STUDENT NOTES:

□ Command Files



COMMAND FILES

Comparison

HP 64000-UX	HP-UX
- non executable file	- executable file
- run within HP 64000-UX application	- run from HP-UX prompt
- created within application or by editor	- created by editor
- can call HP-UX shell script	- can call HP 64000-UX command files (example: page 7-10 of User's Guide)
- parameters can be "passed" or "positional"	- parameters must be "positional" unless code is written for prompting

OVI 7.50 © 1987 Hewlett-Packard Company

MANUAL: page 7-9 HP 64000-UX User's Guide

NOTES:

The chart above lists the major differences between HP 64000-UX command files and HP-UX shell scripts.

Two of the most important points to remember about shell scripts and command files are:

- command files can call HP-UX shell scripts and HP-UX shell scripts can call command files. A example of this is shown on page 7-10 of the User's Guide. You should take a few minutes to look at this example.

- command files will prompt the user for unknown parameters where HP-UX shell scripts will return an error if all the parameters are not specified.

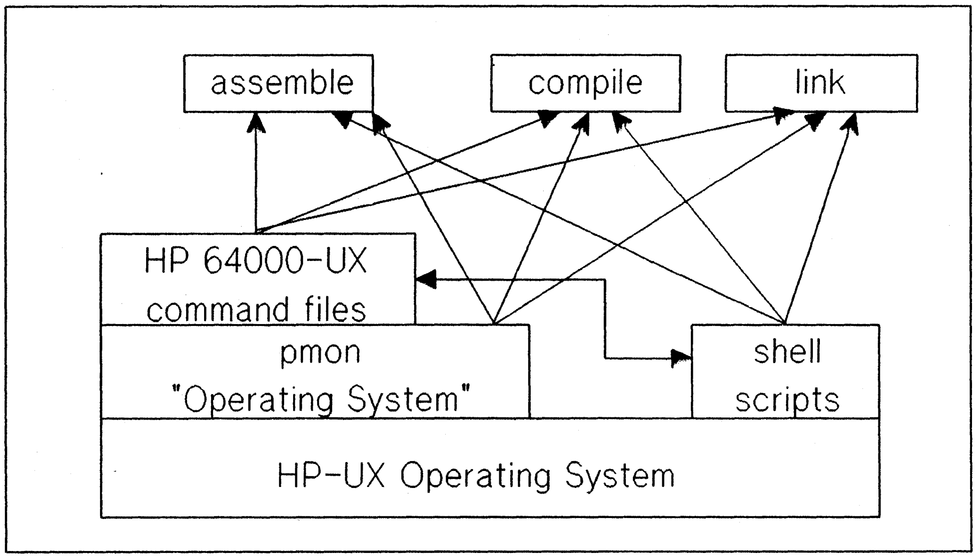
□ Command Files

STUDENT NOTES:

□ Command Files

COMMAND FILES

Graphical Example



OV1 7.55 © 1987 Hewlett-Packard Company

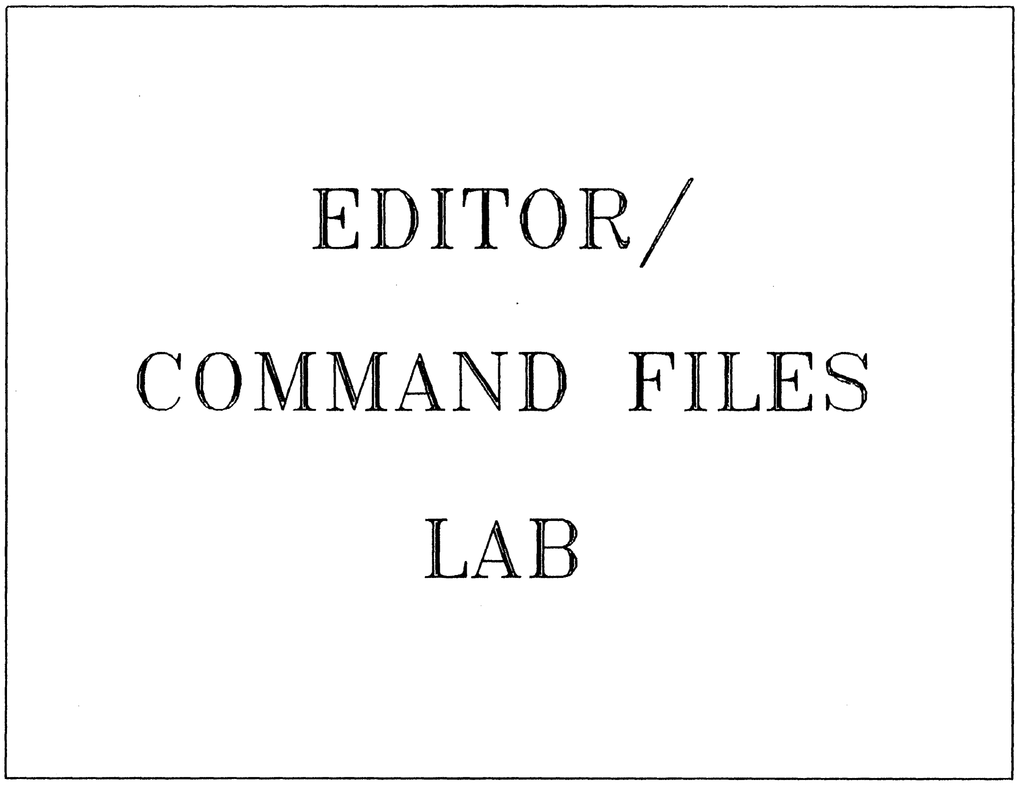
NOTES:

Pmon functions as a type of Operating System for the 64000-UX products. It resides on top of the HP-UX Operating System and gives the user a friendly and learning interface to HP-UX and 64000-UX.

This diagram represents the abilities of 64000-UX command files to call HP-UX shell scripts and vice/versa.

You will notice that the diagram also makes and important point about 64000-UX commands. For the most part, 64000-UX commands also exist as HP-UX commands, but are only mapped as softkeys to make the software designer's job much more efficient.

STUDENT NOTES:



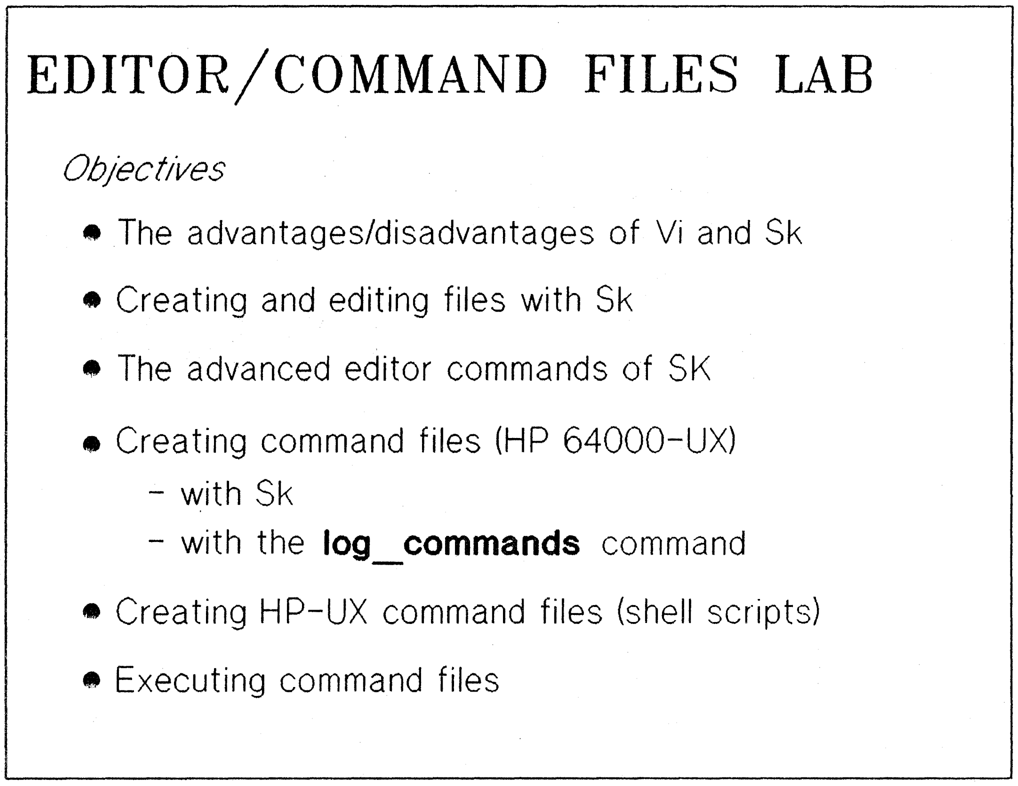
EDITOR/

COMMAND FILES

LAB

ED1 1.05

Editor/Command Files Lab



EDITOR/COMMAND FILES LAB

Objectives

- The advantages/disadvantages of Vi and Sk
- Creating and editing files with Sk
- The advanced editor commands of SK
- Creating command files (HP 64000-UX)
  - with Sk
  - with the **log_commands** command
- Creating HP-UX command files (shell scripts)
- Executing command files

ED1 1.10 © 1987 Hewlett-Packard Company

STUDENT NOTES:

```
# set up the search path
# set up the terminal
# set up the shell environment
# set up the shell variables
```

HP 64000-UX Editor/Command Files Lab

Objectives:

```
    * advantages/disadvantages of Vi and Sk
    * creating and editing files with Sk
    * advanced editor commands of Sk
    * creating command files
         * with Sk
         * with the "log_commands" command
    * creating HP-UX command files (shell scripts)
    * executing command files
```

Each of the sections of the lab are supplied with a box. This box should be checked by the student when the tasks or suggestions of each of these sections are completed.

Accessing the Editors (Vi and Sk)

[ ] Both of these editors have the same syntax for starting an edit session; either creating a new file or editing a existing file. Starting the editors is as easy as:

```
vi                    starts the vi editor
vi <filename>         starts the vi editor to edit <filename
sk                    starts the sk editor
sk <filename>         starts the sk editor to edit <filename
```

The "edit" softkey in pmon is defined to be one of these two editors. Enter the pmon software and touch the edit softkey, but do not hit the return key. Which editor is defined for your environment?

[ ] The edit softkey is defined by the shell variable "EDITOR". This default can be changed in the user's ".profile" file. Use the Sk editor to modify the EDITOR variable. Hint: the Sk editor is located in "/usr/hp64000/bin" and the name of the command is "sk". End out of the edit session using the "end" softkey and execute the following command:

```
. .profile
```

Once again enter pmon and begin to start an edit session. The "edit" softkey should now be mapped to the Sk editor.

You have already edited a file with Sk. Did you have trouble? Or was the softkey driven editor easy to use?

☐ Editor/Command Files Lab

The Vi editor does have its advantages. First it is shipped with HP-UX where the Sk editor is available as an option. Second, the access time and response time of Vi is slightly better than that of Sk. But, the major difference is that Sk is simple to use, even the first time as compared to a cryptic editor like Vi. It is highly recommended that you use the Vi Reference Card to help you get started.

Exiting the Editors

[ ] Once you have gained access to an editor, it is wise to learn how to save a file and end the edit session.

[ ] Enter the Vi editor. Use the Vi Reference Card to determine how to enter the following sentence:

```
      The slow brown fox ran into the lazy dog!
```

Then save this new file with the following command:

```
      <esc> <esc> :wq lazy.dog
```

The two <esc>'s get the editor into its command mode and the ":wq lazy.dog" tells the editor to write the changes, quit the editor and save the contents in a file called lazy.dog.

[ ] You have already had experience ending out of the Sk editor, but since you noticed the expression in "lazy.dog" is wrong, correct it using the Sk editor to:

```
      The quick brown fox jumped over the lazy dog!
```

The Sk Editor

[ ] Since the Vi editor was covered in the first day of this course, equal time must be given to Sk. Have you noticed that many of the same features of pmon are also inherent in the Sk editor. Many of these include the use of softkeys, the same basic display screen, closely related STATUS lines, command completion and command recall.

[ ] As an exercise, print a copy of your .profile file with the following command:

```
      lp .profile
```

Then use the printout as a template to create a new file called "myprofile" which has the exact contents as your .profile.

☐ Editor/Command Files Lab

[ ] Take 20 to 30 minutes to learn as many features of the Sk editor as possible. Remember, the same help facilities exist in the Sk editor as did in pmon:

```
      "help" command
      syntax directed softkeys
```

Another good file to play with can be created with the following command:

```
      banner I love Sk > editor.play
```

Learn the uses of the following commands:

REVISE	delete	end	save
INSERT	<LINE+->	extract	retrieve
replace	find	copy	renumber
help	save		

One of the most powerful features of the Sk editor is the ability to do different types of range editing.

If time permits and you like challenges:

repeat	range	while	insert
list	join	split	WHATCHAR
edit	merge	log	

IMPORTANT!!! The absolute best time to ask questions about a new product is during a lab like this.

Creating Command Files

[ ] Make sure you exit the Sk editor and return to the pmon software.

[ ] The exercise which follows will give you more practice with the Sk editor and also introduce you to command files. It will also show you that command files are available for both Sk and pmon. Exit any edit session you may be working on and enter pmon. Begin logging commands to a command file by using the "log" softkey. Name the command file "edit.monitor". Read the help messages if necessary or ask for help.

☐ Editor/Command Files Lab

[ ] From pmon issue the command to copy the file "$HOME/emul/mon_68000.S" to a file named "$HOME/new_monitor". The following editing of the "new_monitor" file should be completed.

[ ] Using the Sk editor softkeys, replace every occurrence of the string "MONITOR_CONTROL" with the string "MONITOR_CNTRL".

[ ] Uncomment the "TRAP #5" software breakpoint vector and its appropriate ORG statement.

[ ] Now this is a challenge. In the file, there are many lines which consist of nothing but "*"'s. Replace each "*" with a "#" in the 25th through the 50th columns of these lines. In essence, replace:

```
**********************************************************************
```

with:

```
************************##########################********************
```

HINT: use the range and replace commands to make this easy!

[ ] When that task is complete, end out of the editor and save your changes. Then issue the command to pmon to close the log file.

[ ] Edit the command log file (edit.monitor) to see what it looks like. You should notice that it logged every command you made into the file whether it was an error or not. If you ever plan to use a command file that you create in this way, you MUST edit the unwanted commands out of the file or they will be executed just will the wanted commands. Edit the command file so that each task specified above can be completed automatically. To test your command file, make sure the command to copy the original file to your home directory and the command to enter the Sk editor are still in your command file and execute the command file by typing "edit.monitor" on the pmon command line. Verify its proper operation and fix any bugs which may arise. NOTE: be sure to include the command to exit the Sk editor when the command file is finished. Also you can use the "wait" command in the command file to help verify its proper operation by pausing the execution of the command file.

[ ] Command files can also be created by entering the pmon or sk commands into a file with an editor. The only gotcha with this method is that you do not have the helping hand of syntax directed softkeys to help keep the errors to a minimum.

☐ Editor/Command Files Lab

[ ] One important point to keep in mind concerning 64000-UX command files, is that they can only be run from within the program, i.e. within pmon or Sk. These files do not have the ability to be run as separate executable files, i.e. NEVER change the permissions on a 64000-UX command file to make it executable.

[ ] A more efficient and productive use of command files will be introduced during the emulation section of this course. This lab is meant only to introduce the methods and concepts of creating command files. In reality, they can save large amounts of time by eliminating the need to enter repetitive commands over and over again.

[ ] As an exercise in parameter passing for the 64000-UX command files, copy the example from page 10 of the Command Files section in the student workbook (comments optional). Get the command file to work by practicing on different words in the "new_monitor" file.

[ ] As a final exercise, create a command file which will modify a specified file and add 5 blank columns to the leftmost portion of the file. This will be helpful in changing the margins of a file if it needs to be printed out. HINT: range command and replace command.

[ ] Test this command file and make sure it works properly.

HP-UX Command Files (shell scripts)

[ ] Use the editor of your choice to create a file called "find.it". In that file enter the following:

```
# usage:   find.it  <string>  <filename>
grep $1 $2
```

Save the file.
The "$1" and "$2" are considered positional parameters. These parameters become part of the command when it is executed, just like the filenames of the "cp" command are necessary.

[ ] Execute the shell script to search for your username entry in the /etc/passwd file.
Example:

```
find.it root /etc/passwd
```

☐ Editor/Command Files Lab

[ ] As you noticed, you were flagged with an error telling you that the "execute permission was denied". This is the big difference between HP-UX shell scripts and 64000-UX command files. The HP-UX shell scripts must have their execute permissions set to be of any use.

Use the following command to set the permissions properly:

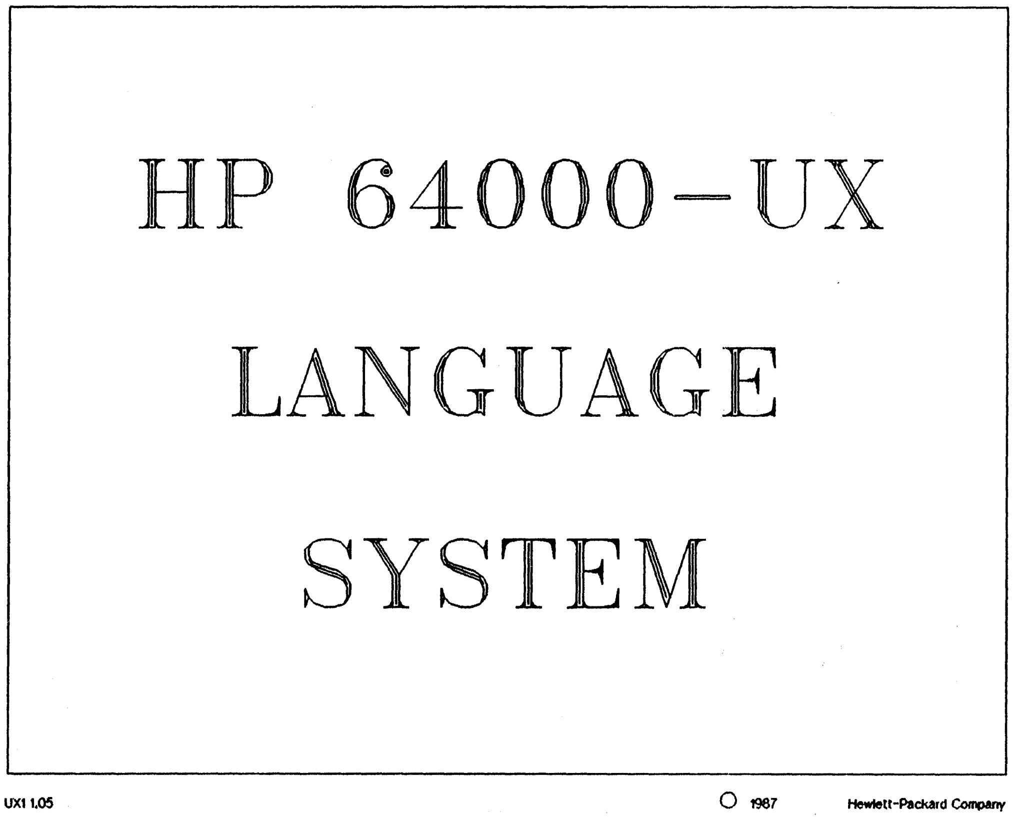
```
chmod 777 find.it
```

[ ] Re-run the shell script to search for your login name in the /etc/passwd file. Do not forget to include the positional parameters.

Not only can shell scripts have parameters, but so can the 64000-UX command files. This feature opens up the flexibility of command files so that their functionality is only limited by an individuals resourcefulness.

[ ] A perfect example of a shell script is your ".profile" file which resides in your HOME directory. Do a listing of the permissions of this file to verify that it is an acceptable shell script.

□ HP 64000-UX Language System



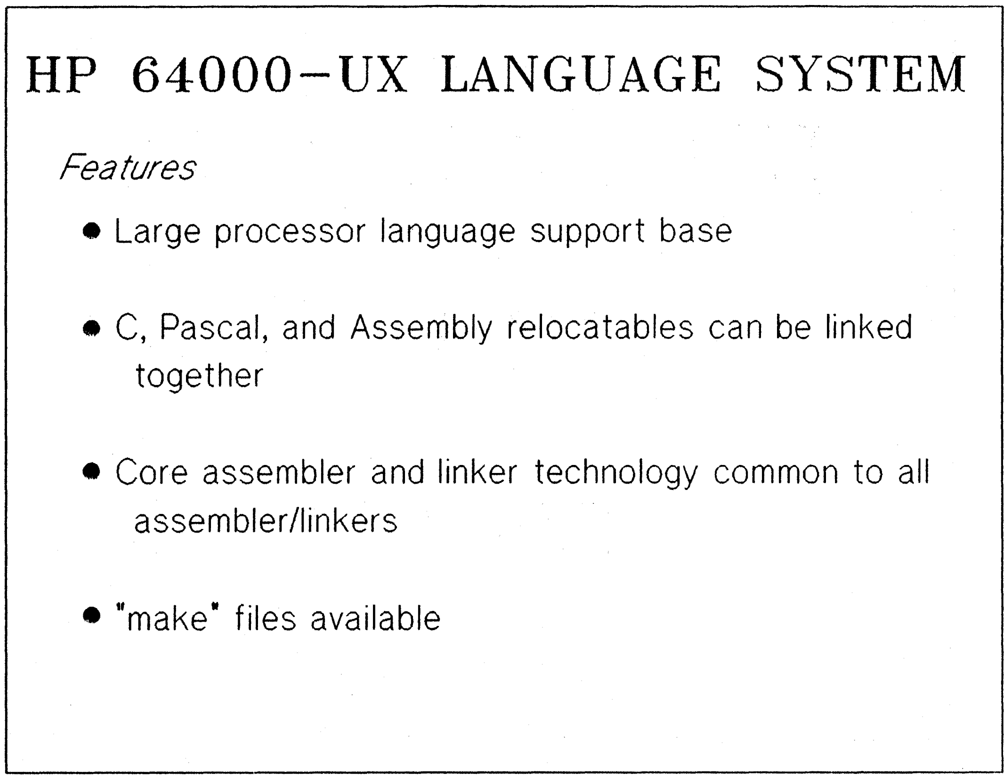
HP 64000–UX

LANGUAGE

SYSTEM

UX1 1.05

□ HP 64000-UX Language System



HP 64000-UX LANGUAGE SYSTEM

Features

- Large processor language support base
- C, Pascal, and Assembly relocatables can be linked together
- Core assembler and linker technology common to all assembler/linkers
- "make" files available

UX1 1.10 ○ 1987 Hewlett-Packard Company

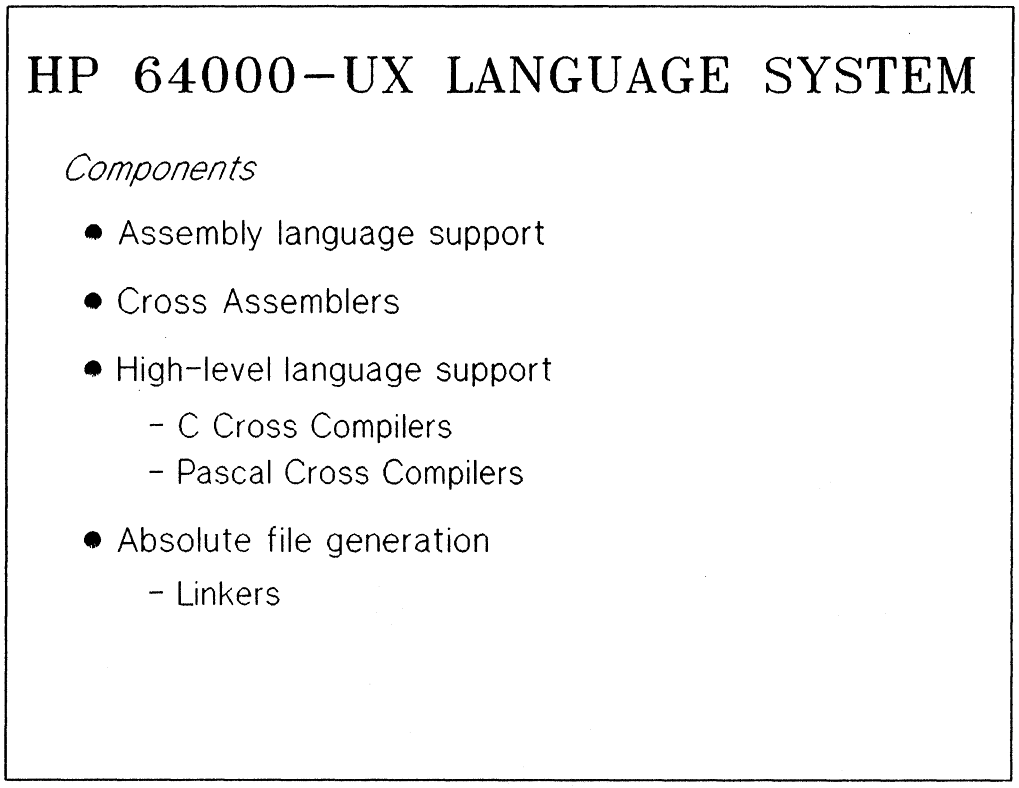
NOTES:

C and Pascal compilers and assemblers are available for a large number of processors and processor families.

The HP 64000-UX Language System assemblers and compilers generate relocatables with the same file format. These relocatables can therefore be linked together to form one absolute file.

makefiles: same function as command files but only perform the parts of assembling / compiling / linking which are needed. The "make" command uses file time stamping to determine if a given relocatable is up-to-date. This utility is also used for source code control.

STUDENT NOTES:



HP 64000–UX LANGUAGE SYSTEM

Components

- Assembly language support
- Cross Assemblers
- High–level language support
  - C Cross Compilers
  - Pascal Cross Compilers
- Absolute file generation
  - Linkers

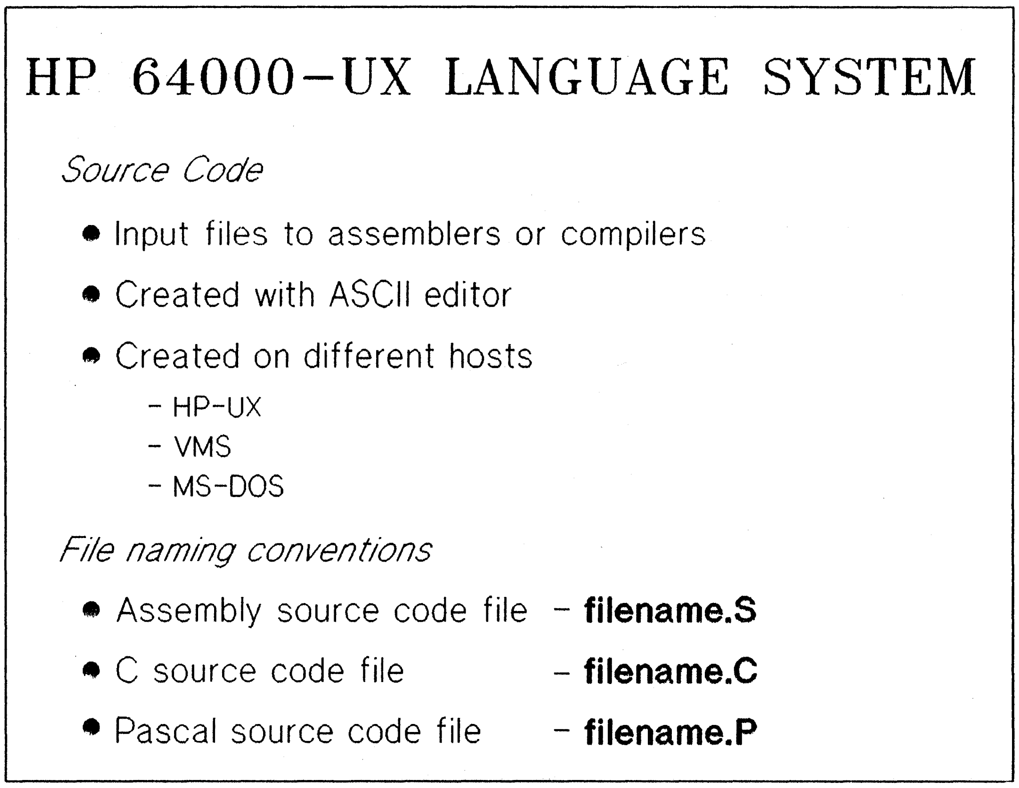
UXI 1.15 ○ 1987 Hewlett-Packard Company

NOTES:

The HP 64000-UX Language System consists of cross assemblers and cross compilers which generate the relocatable files. It also consists of linkers which combine the relocatable files into an absolute file.

The word "cross" refers to language system tools which create files to be used on other hosts. A compiled file to be used on the 9000 series 300 computer would have a "native" compiler.

STUDENT NOTES:



HP 64000–UX LANGUAGE SYSTEM

Source Code

- Input files to assemblers or compilers
- Created with ASCII editor
- Created on different hosts
  - HP-UX
  - VMS
  - MS-DOS

File naming conventions

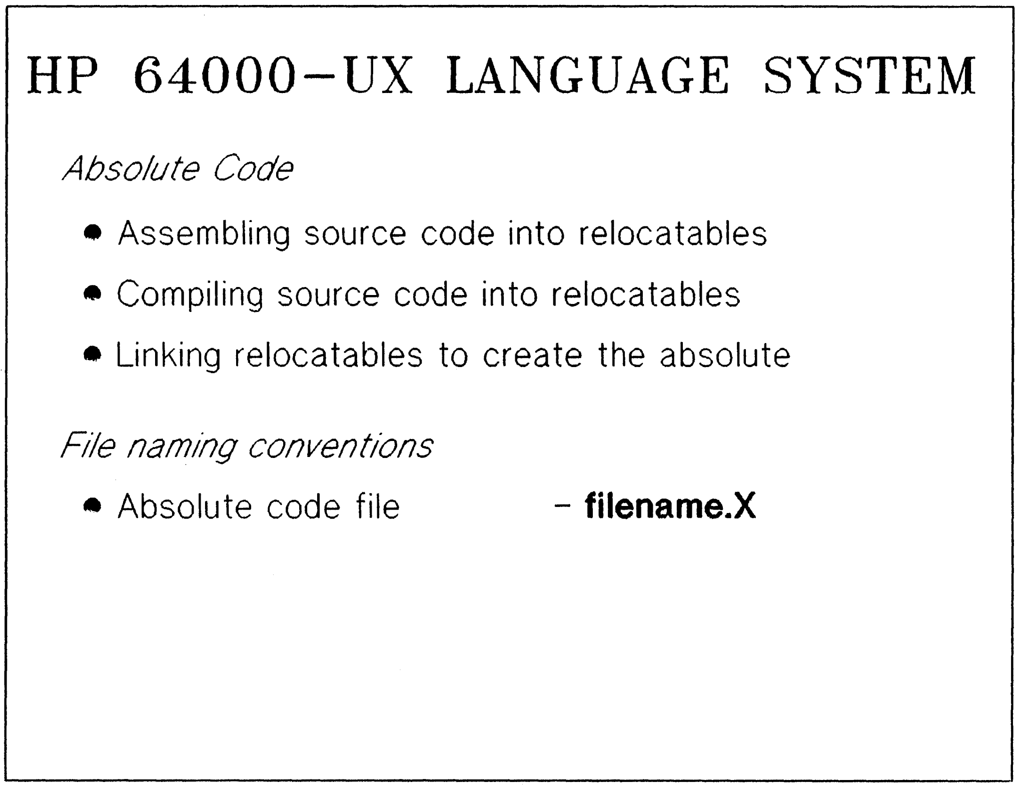
● Assembly source code file	– **filename.S**
● C source code file	– **filename.C**
● Pascal source code file	– **filename.P**

UX1 1.20

NOTES:

The file naming conventions for source code files are strongly recommended, but not necessary.

STUDENT NOTES:



HP 64000-UX LANGUAGE SYSTEM

Absolute Code

- Assembling source code into relocatables
- Compiling source code into relocatables
- Linking relocatables to create the absolute

File naming conventions

- Absolute code file - **filename.X**

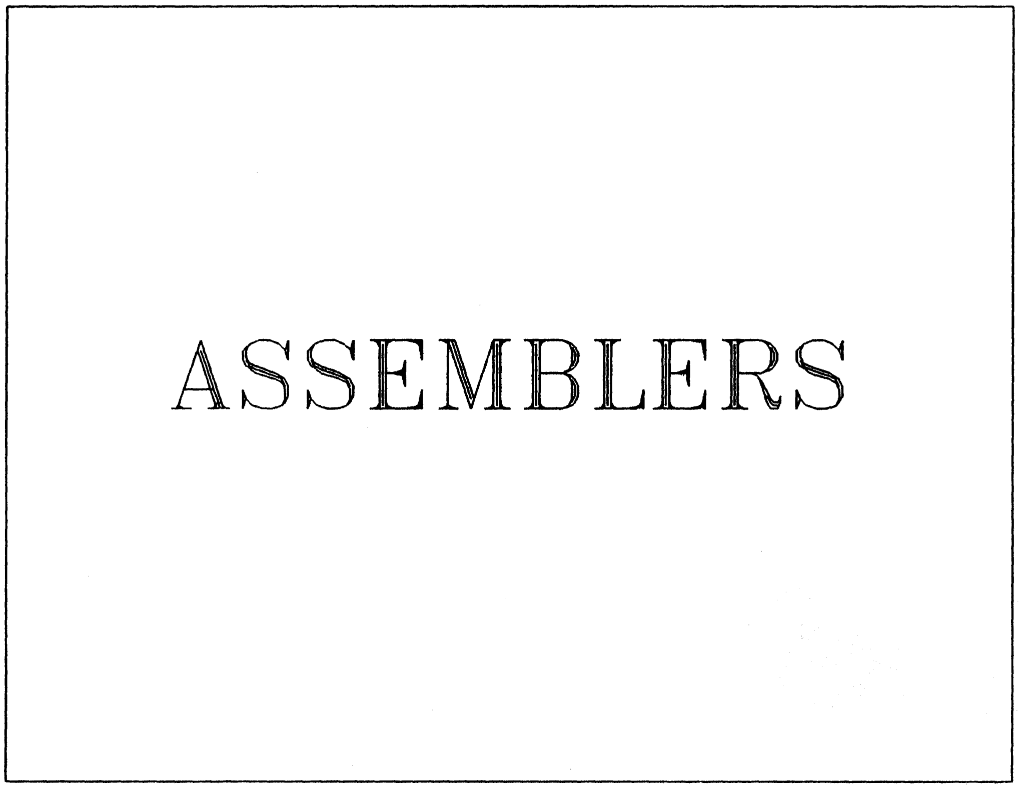
UX1 1.25 © 1987 Hewlett-Packard Company

NOTES:

The file naming convention for the absolute code file is defined by the linker.

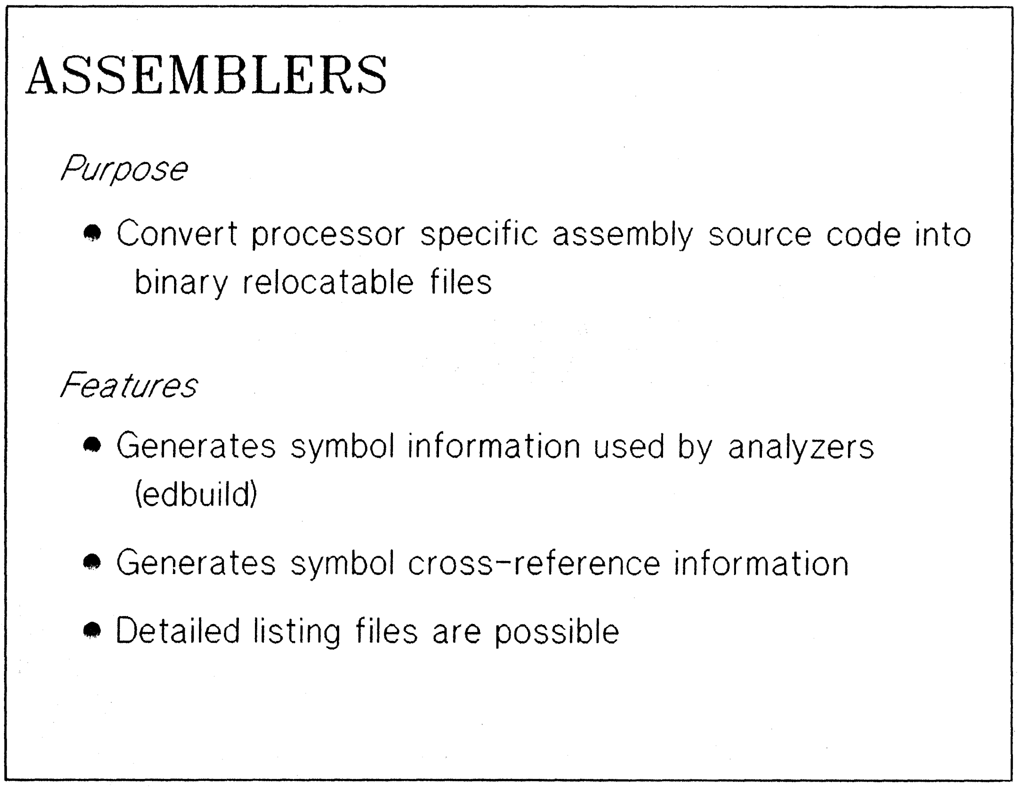
STUDENT NOTES:

□ Assemblers



ASSEMBLERS

AS1 1.05 ○ 1987 Hewlett-Packard Company



ASSEMBLERS

Purpose

- Convert processor specific assembly source code into binary relocatable files

Features

- Generates symbol information used by analyzers (edbuild)
- Generates symbol cross-reference information
- Detailed listing files are possible

ASI 1.10

MANUAL: page 1-2 68000 Cross Assembler/Linker

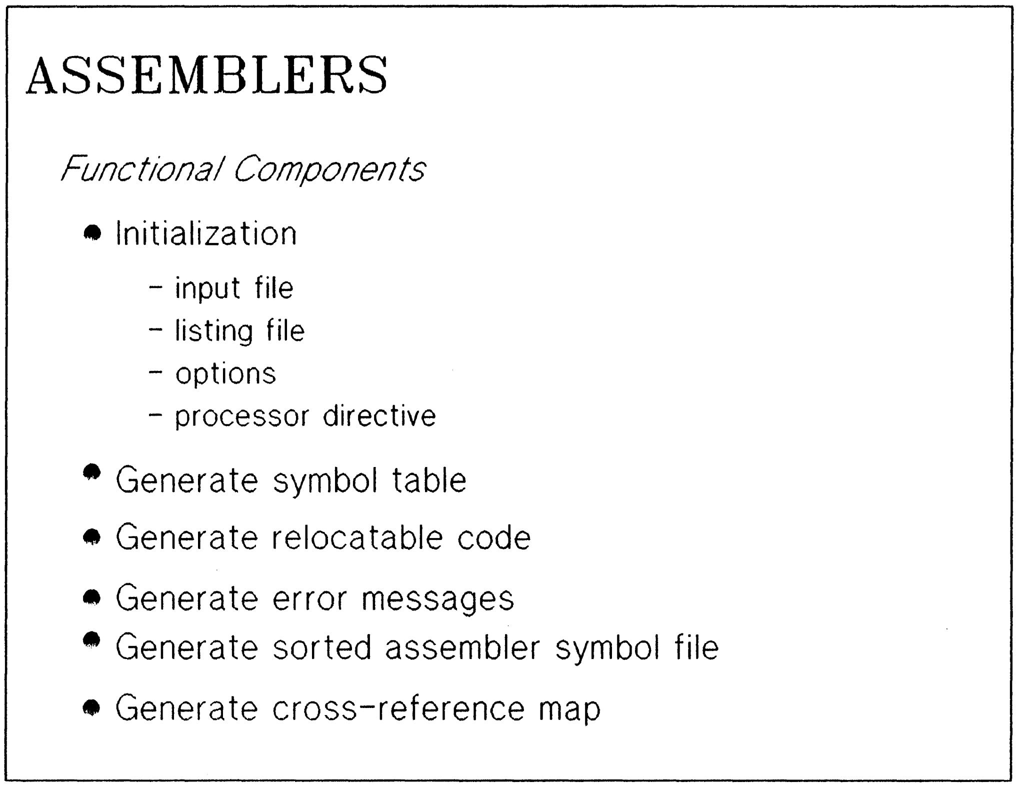
NOTES:

The assembler generates a ".A" file which is used by the edbuild command to create the symbol database file ".Y", which is in turn used by the 64000-UX analyzers and emulators.

Options to the listing files are available and will be explained later in this section.

STUDENT NOTES:

☐ Assemblers



ASSEMBLERS

Functional Components

- Initialization
  - input file
  - listing file
  - options
  - processor directive
- Generate symbol table
- Generate relocatable code
- Generate error messages
- Generate sorted assembler symbol file
- Generate cross-reference map

ASI 1.15 © 1987 Hewlett-Packard Company

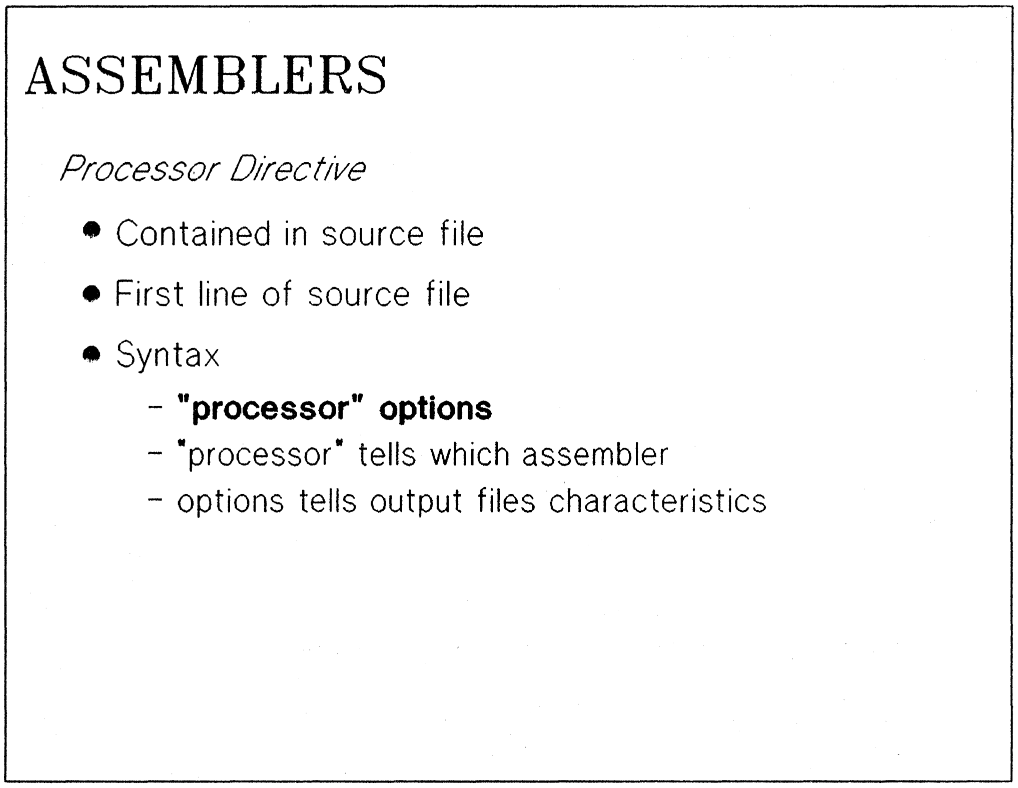
MANUAL: page 2-1 68000 Cross Assembler/Linker

NOTES:

Generating the cross-reference map is only performed if the "xref" option is specified for the listing file.

You can find a good introduction to assembler technology in Appendix A of the 68000 Cross Assembler/Linker manual.

STUDENT NOTES:



ASSEMBLERS

Processor Directive

- Contained in source file
- First line of source file
- Syntax
  - **"processor" options**
  - "processor" tells which assembler
  - options tells output files characteristics

MANUAL: pages 2-3 thru 2-5 68000 Cross Assembler/Linker

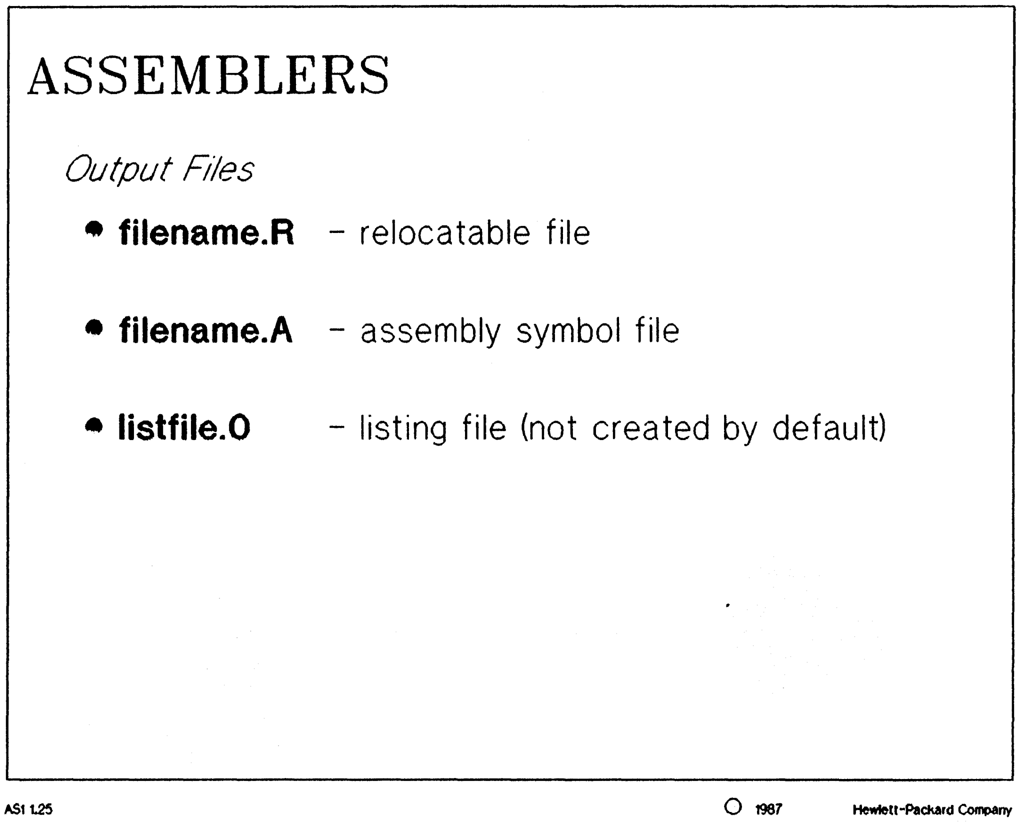
NOTES:

The processor directive for a source file MUST be on the first line of the file, with no blank lines preceding it.

The options which can be specified as part of the processor directive command can also be specified as options to the HP-UX command.

STUDENT NOTES:

☐ Assemblers



ASSEMBLERS

Output Files

- **filename.R** - relocatable file
- **filename.A** - assembly symbol file
- **listfile.O** - listing file (not created by default)

AS1 1.25 © 1987 Hewlett-Packard Company

MANUAL: pages 2-2 thru 2-3 68000 Cross Assembler/Linker

NOTES:

filename.R: this file is the relocatable for the source code which was assembled. It has the same base filename as the source file and contains the assembled object code.

filename.A: this file contains the symbol and line number information of the relocatable file. It is used by the edbuild command, which will be discussed later.

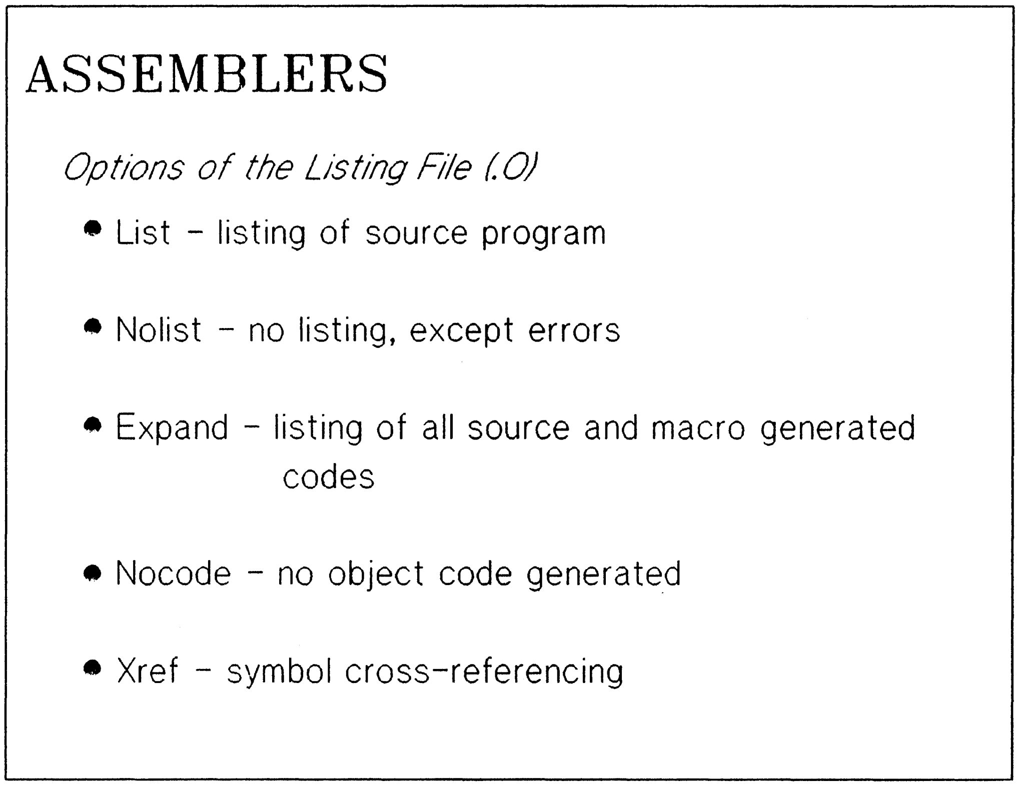
listfile.O: this file is optionally created by the user and contains the assembly code with offset addresses specified for the symbols and code. It can also contain a symbol cross reference table depending on the options specified.

The ".R" and ".A" files are created by default by the assembler, whereas the ".O" file is not created by default. The ".O" convention is the standard for any type of listing file. It is recommended that this convention is used, but is not necessary.

CAUTION: if the ".O" convention is used, be careful not to name the assembler or compiler listing file the same as the linker listing file, or the assembler or compiler listing file will be overwritten!

STUDENT NOTES:

□ Assemblers



ASSEMBLERS

Options of the Listing File (.O)

- List - listing of source program
- Nolist - no listing, except errors
- Expand - listing of all source and macro generated codes
- Nocode - no object code generated
- Xref - symbol cross-referencing

AS1 L30

MANUAL: pages 2-6 thru 2-9 68000 Cross Assembler/Linker

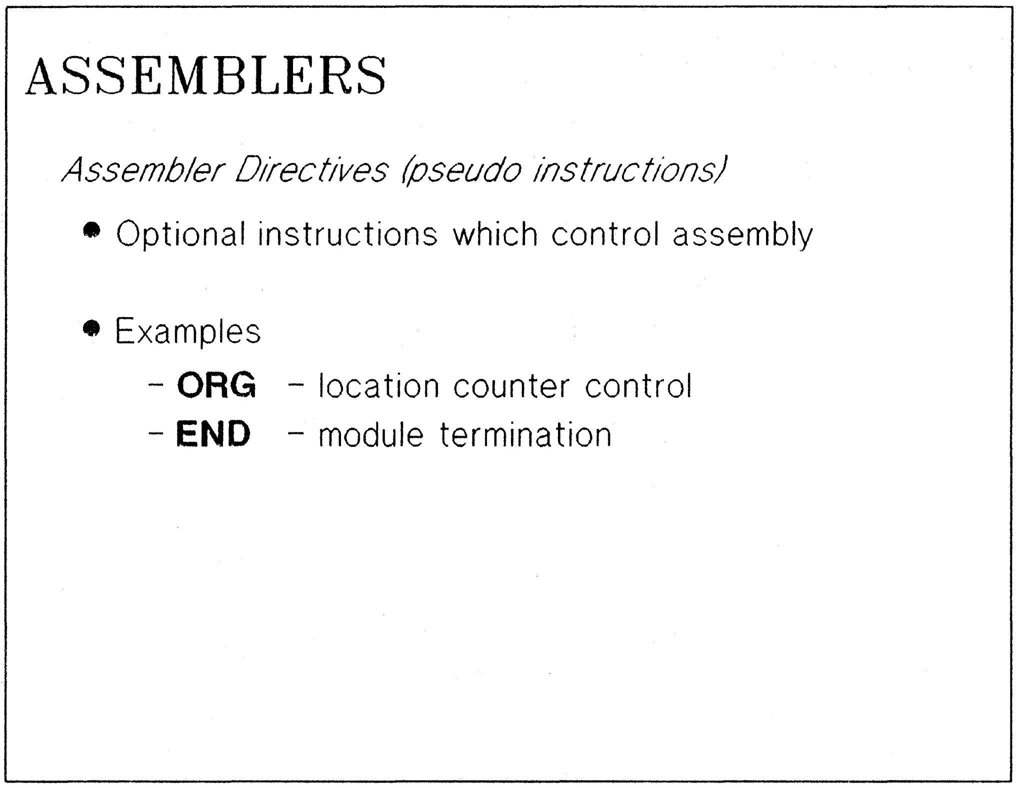
NOTES:

Listing files are not necessary, but HIGHLY RECOMMENDED!

The options to the assembler can be specified in the source code file as options to the processor directive (page 2-3) or as options to the HP-UX command (page 2-5) which are mapped as softkeys.

STUDENT NOTES:

☐ Assemblers



ASSEMBLERS

Assembler Directives (pseudo instructions)

- Optional instructions which control assembly
- Examples
  - **ORG** – location counter control
  - **END** – module termination

MANUAL: pages 5-1 thru 5-3 68000 Cross Assembler/Linker

NOTES:

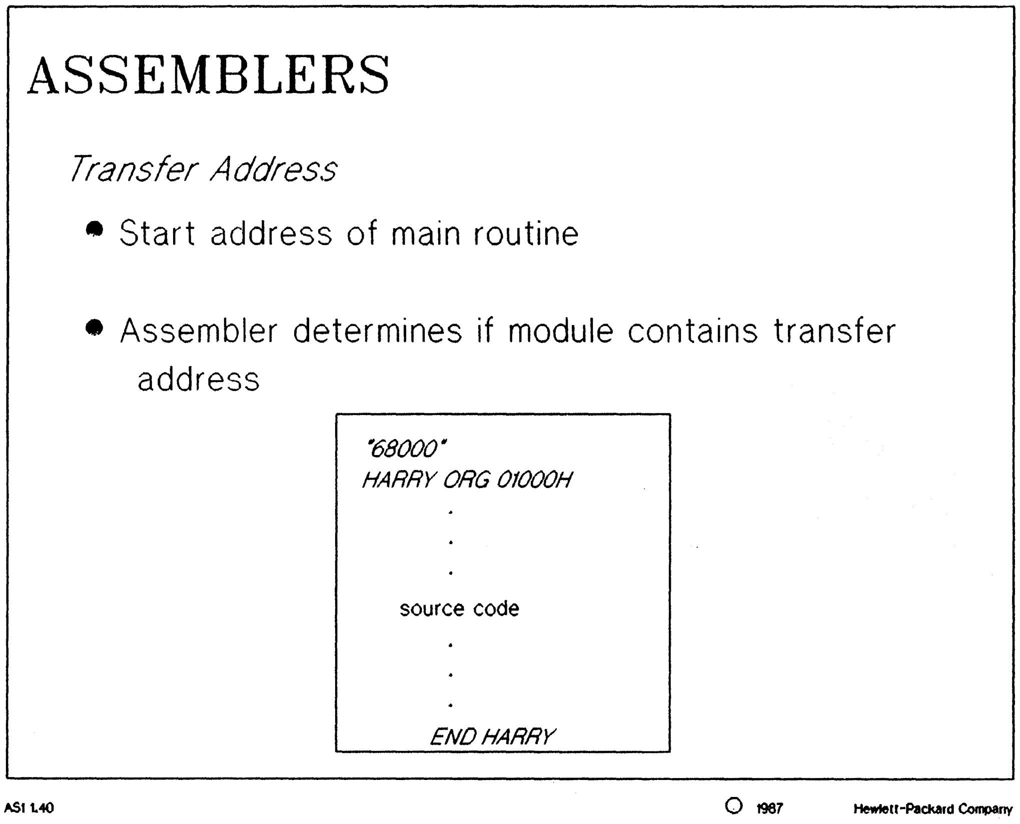
The two pseudo instructions listed are probably used more than any others.

ORG: (page 5-20) this instruction is used to specify an absolute address for the code listed after it. It sets the location counter to the address specified in the operand field of the source code.

END: (page 5-9) this instruction is used to show the logical end of a program module. It is optional, and if specified with a label will define a transfer address offset for that module. Transfer addresses will be covered on the next slide.

STUDENT NOTES:

Assemblers



ASSEMBLERS

Transfer Address

- Start address of main routine
- Assembler determines if module contains transfer address

```
"68000"
HARRY ORG 01000H
        .
        .
        .
    source code
        .
        .
        .
     END HARRY
```

AS1 1.40 © 1987 Hewlett-Packard Company

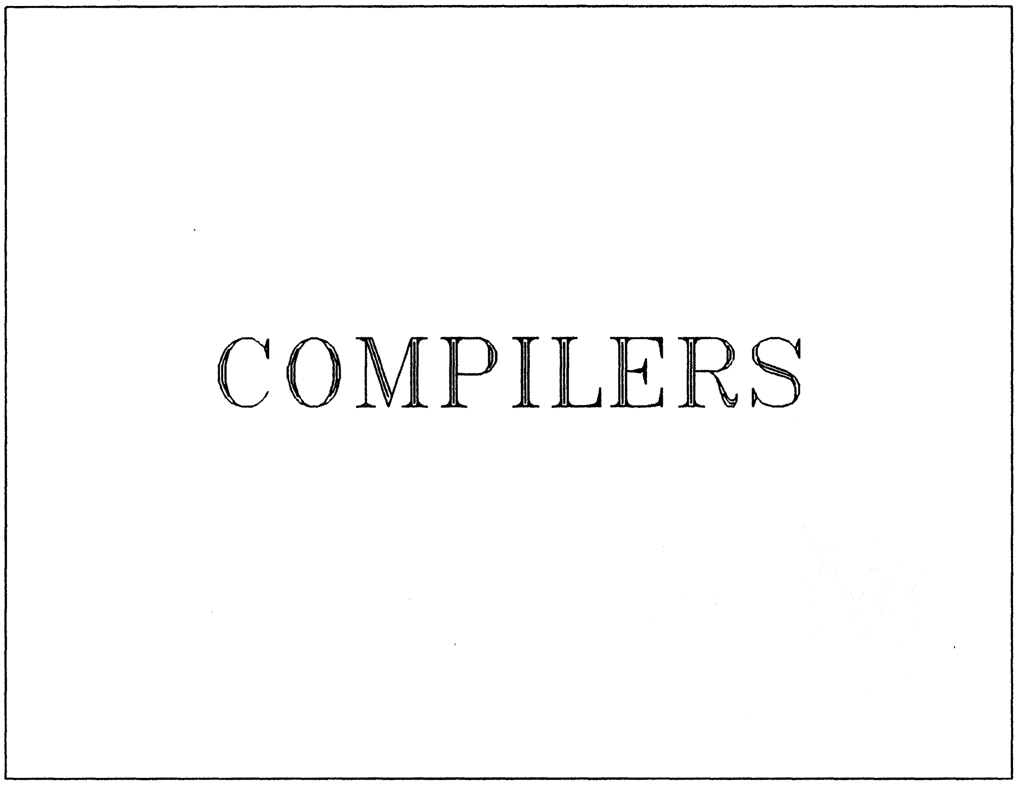
NOTES:

transfer address: this is defined in one of the relocatable files of a program to be the starting address of the main program.

The label "HARRY" tells the assembler that this module will contain the transfer address. Only one transfer address is allowed for the relocatable files which will be linked together to form the absolute. For the example shown, the transfer address will be defined as 1000H.

Specifying an "END" with no label or not including the "END" directive will not define the transfer address. It will default the transfer address to 0000H until it is defined by another relocatable file.

STUDENT NOTES:



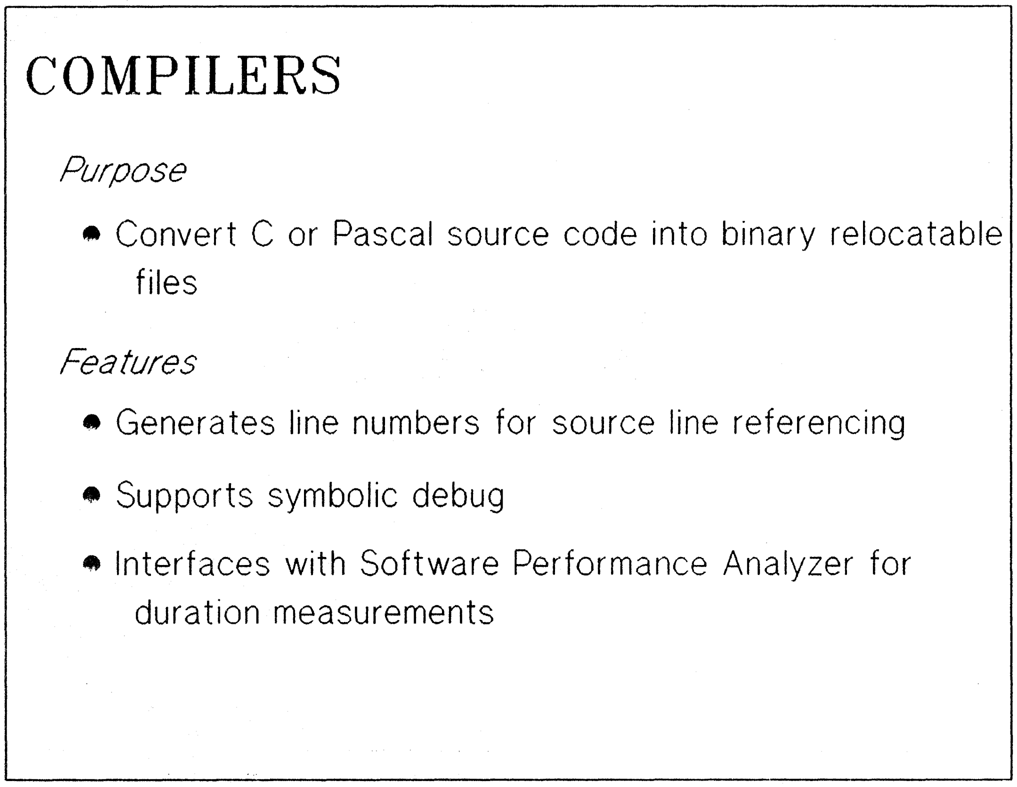
COMPILERS

C01 1.05

NOTES:

The C compiler language manual will be used for this section, but the compiler information presented here applies to both the C compiler and the Pascal compiler, except where noted.

□ Compilers



COMPILERS

Purpose

- Convert C or Pascal source code into binary relocatable files

Features

- Generates line numbers for source line referencing
- Supports symbolic debug
- Interfaces with Software Performance Analyzer for duration measurements

CO1 1.10 © 1987 Hewlett-Packard Company

MANUAL: pages 2-1 thru 2-3 68000 C Cross Compiler

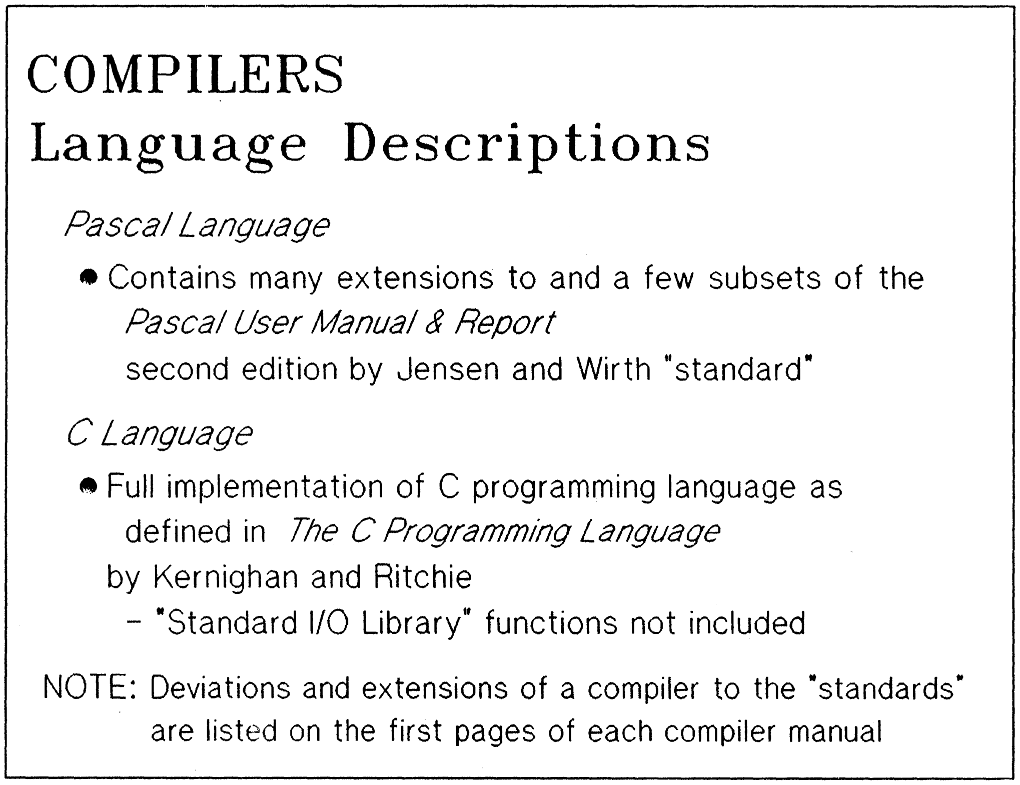
NOTES:

Line numbers are generated by the compiler and contained with local symbols in the ".A" file.

symbolic debug: using symbols for physical address references. An example would be: "run from <symbol>"

The compiler also generates the information necessary for the software performance analyzer to perform duration measurements on modules of code.

STUDENT NOTES:



COMPILERS
Language Descriptions

Pascal Language

- Contains many extensions to and a few subsets of the *Pascal User Manual & Report* second edition by Jensen and Wirth "standard"

C Language

- Full implementation of C programming language as defined in *The C Programming Language* by Kernighan and Ritchie
  - "Standard I/O Library" functions not included

NOTE: Deviations and extensions of a compiler to the "standards" are listed on the first pages of each compiler manual

C01 1.12 © 1987 Hewlett-Packard Company

NOTES:

The HP 64000-UX Pascal Cross Compilers implement an extension of the Pascal programming language which is specifically designed and optimized for microprocessor development. Although closely aligned with the Pascal language as defined in "Pascal User Manual and Report - Second Edition" by Kathleen Jensen and Niklaus Wirth (Springer-Verlag, 1976), the HP 64000-UX System Pascal language contains many extensions and a few subsets of the Jensen and Wirth "standard"

□ Compilers

The HP 64000-UX C Cross Compilers provide a full implementation of the C programming language as defined in "The C Programming Language" by Brian W. Kernighan and Dennis M. Ritchie (Prentice-Hall, 1978) and the supplement published November 15, 1978. Hewlett-Packard has also extended the C language to improve its utility as a tool for microprocessor system programming and to make optimum use of the HP-UX and HP 64000-UX software development tools.

The "standard I/O library" functions (printf, getchar, etc) are not included. External references are generated by the compiler for these functions.

STUDENT NOTES:

☐ Compilers

COMPILERS

Functional Description

- Optional preprocessor – facilities to alter source code

- Reads source code
- Creates symbolic language (IDS)

- Reads IDS
- Generates cryptic assembly language invented
  by HP

- Processor specific code generator creates
  relocatable

CO1 1.15 ○ 1987 Hewlett-Packard Company

MANUAL: page 5-2 68000 C Cross Compiler

NOTES:

The preprocessor can be used for the following:
- handling include files
- handling user-defined macros
- defining constants
  ex. test flags
    If <test flag> = 1 converts to
    If <test flag> = true

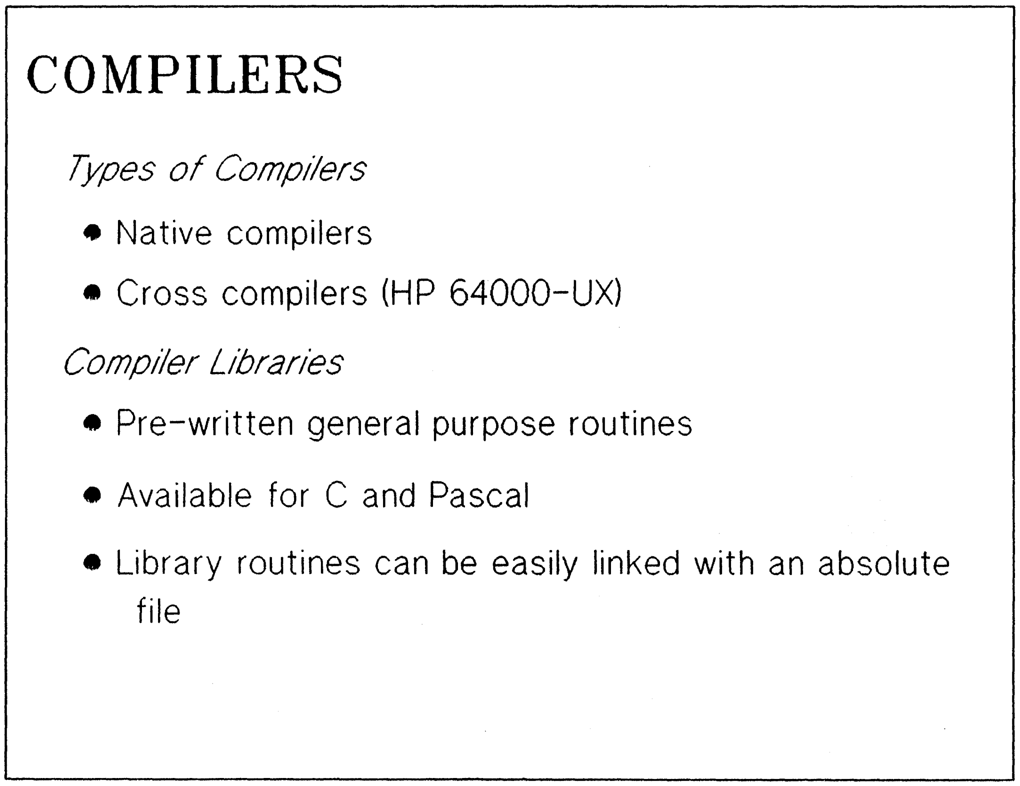
IDS (intermediate data structure): a symbolic language which is read by the compiler to generate a cryptic assembly language for a given processor.

Reading the source code and generating the IDS is high level language dependant. This part of the compiler is the same for all C compilers. This part of the compiler is also the same for all Pascal compilers, yet it is different from the C compilers.

Reading the IDS, generating the cryptic assembly language, and generating the relocatable are processor specific functions; yet are the same in the C and Pascal compilers.

STUDENT NOTES:

☐ Compilers



COMPILERS

Types of Compilers

- Native compilers
- Cross compilers (HP 64000-UX)

Compiler Libraries

- Pre-written general purpose routines
- Available for C and Pascal
- Library routines can be easily linked with an absolute file

CO1 1.20 © 1987 Hewlett-Packard Company

NOTES:

`native compiler:` this is a compiler that generates code for the host it runs on, i.e. the 68020 C compiler for the 9000 series 300 computers.

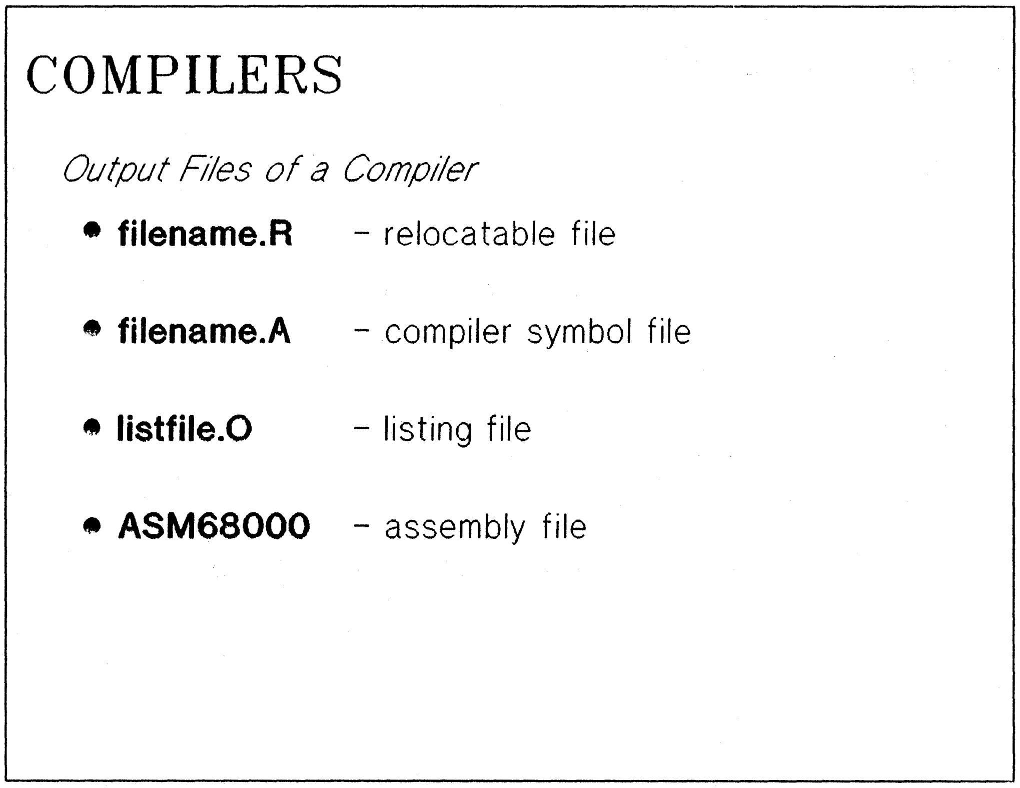
`cross compiler:` this is a compiler that can generate code on one host and be used on another host, i.e compiling 68000 C code on a 68020 based 9000 series 300 computer.

MANUAL: pages 4-1 thru 4-33 68000 C Cross Compiler

`compiler libraries:` pre-written routines which can be linked with the user's relocatable code to form the absolute code.

STUDENT NOTES:

☐ Compilers



COMPILERS

Output Files of a Compiler

- **filename.R** - relocatable file
- **filename.A** - compiler symbol file
- **listfile.O** - listing file
- **ASM68000** - assembly file

CO1 1.25 © 1987 Hewlett-Packard Company

MANUAL: pages 2-2 thru 2-3 68000 C Cross Compiler

NOTES:

The ".R" and ".A" files are created as the default of the compiler. The ".O" and the "ASM68000" files are created only as specified by the user at compile time.

filename.R: this is the binary relocatable file which contains the offset addresses of the source code.

filename.A: this file contains the line number and local symbol information for the module.

ASM68000: this file contains the 68000 assembly language which was generated by the high level language compiler. This file is only created if the user specifies its creation as an option to the compiler.

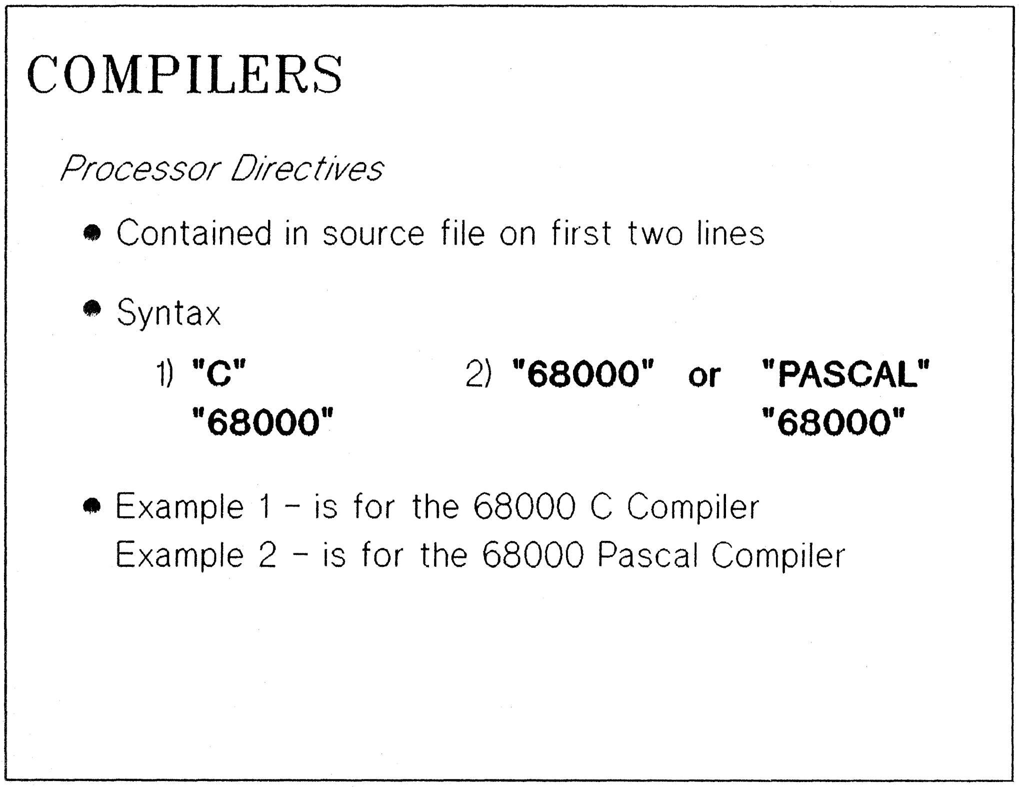
MANUAL: pages 5-2 thru 5-4 68000 C Cross Compiler

listfile.O: this is the optional listing file. The same options available in the 68000 assembler are also available in the compiler, such as the cross reference option for symbols. An example of this is shown on page 5-4 of the manual.

The same caution applies to the compiler listing file as did the assembler listing file.

STUDENT NOTES:

☐ Compilers



COMPILERS

Processor Directives

- Contained in source file on first two lines
- Syntax

```
1) "C"                2) "68000"  or  "PASCAL"
   "68000"                            "68000"
```

- Example 1 - is for the 68000 C Compiler
  Example 2 - is for the 68000 Pascal Compiler

CO1 1.30 © 1987 Hewlett-Packard Company

MANUAL: pages 2-2 thru 2-3 68000 C Cross Compiler

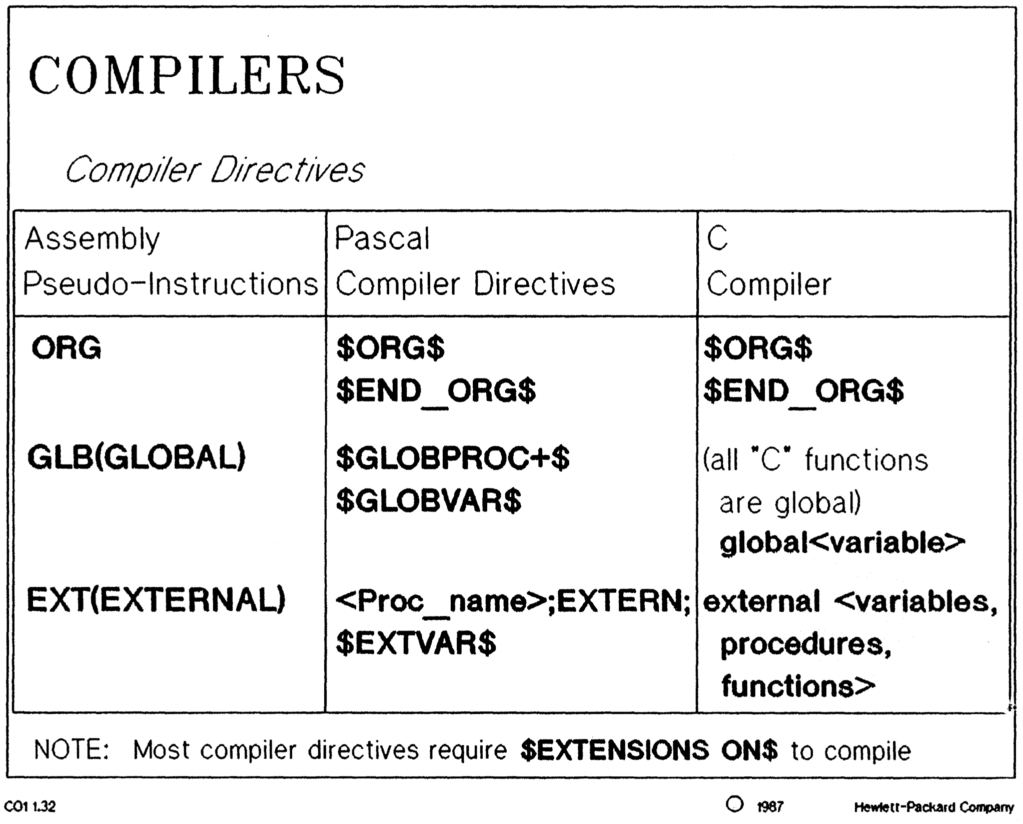
NOTES:

The processor directive information MUST be on the first line or two lines of the source code file. No blank lines are allowed above these statements.

The processor directive for the Pascal compiler can be exactly the same as the directive for the 68000 assembler.

STUDENT NOTES:

□ Compilers



COMPILERS

Compiler Directives

Assembly Pseudo-Instructions	Pascal Compiler Directives	C Compiler
ORG	**ORG** **END_ORG**	**ORG** **END_ORG**
GLB(GLOBAL)	**$GLOBPROC+$** **$GLOBVAR$**	(all "C" functions are global) **global<variable>**
EXT(EXTERNAL)	**<Proc_name>;EXTERN;** **$EXTVAR$**	**external <variables, procedures, functions>**

NOTE: Most compiler directives require **$EXTENSIONS ON$** to compile

C01 1.32 ○ 1987 Hewlett-Packard Company

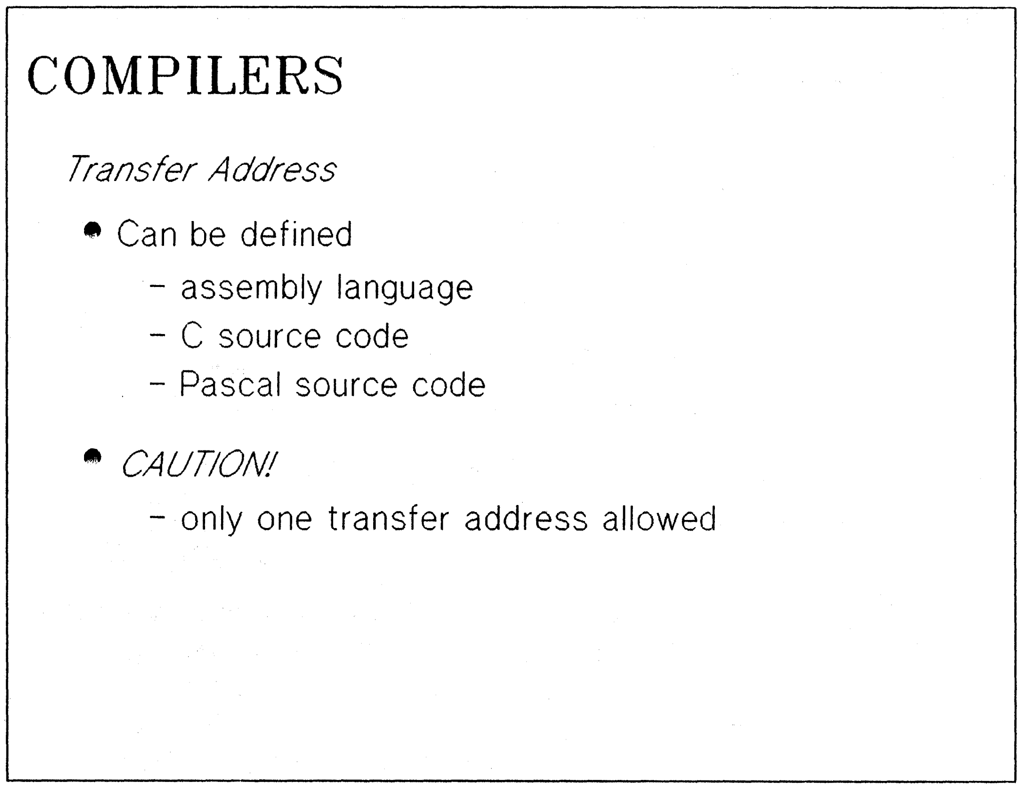
NOTES:

The above slide gives comparisons of the some of the common compiler/assembler directives. Consult the individual compiler/assembler manual for a complete list of these directives.

NOTE: to turn off "$GLOBVAR$" in Pascal use the directive "$GLOBVAR-$" or "GLOBVAR OFF".

STUDENT NOTES:

Compilers



COMPILERS

Transfer Address

- Can be defined
  - assembly language
  - C source code
  - Pascal source code
- *CAUTION!*
  - only one transfer address allowed

C01 1.35 © 1987 Hewlett-Packard Company

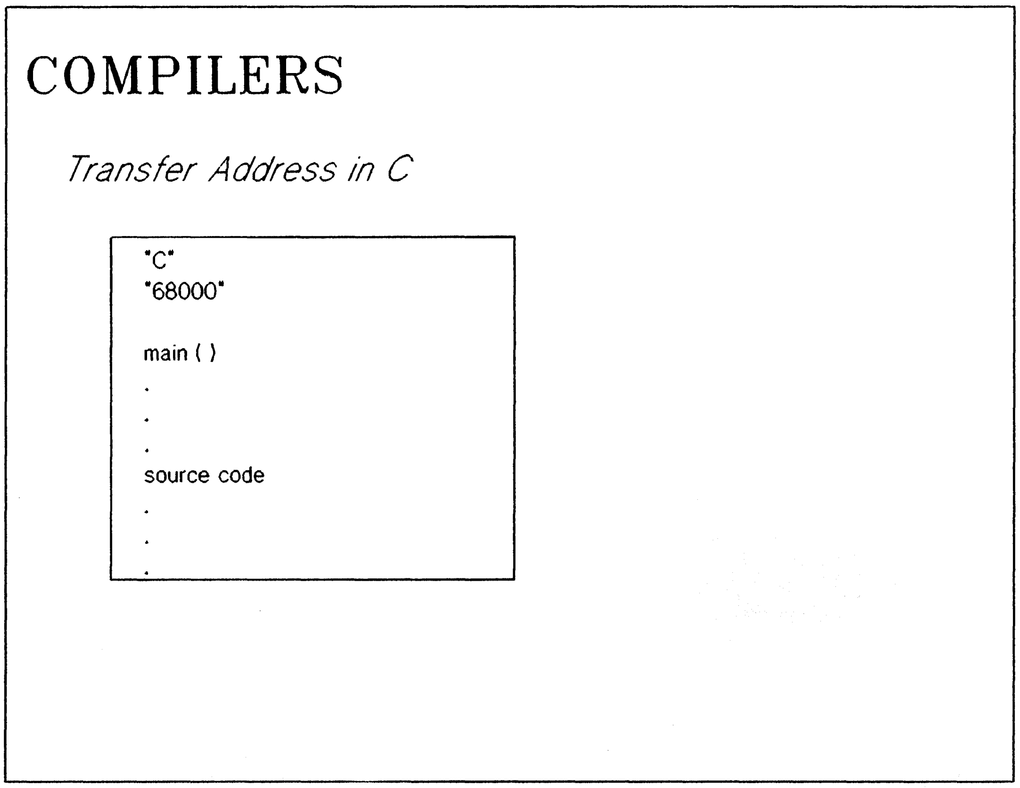
NOTES:

Both the C and Pascal compilers as well as the assembler have a unique method of determining the transfer address for a given source code module. Please make a note of these differences in the next few slides.

Multiple transfer addresses (other than those defaulted to 0000H) will result in a warning when the linker attempts to assign absolute addresses to the various modules.

STUDENT NOTES:

□ Compilers



COMPILERS

Transfer Address in C

```
"C"
"68000"

main ( )
.
.
.
source code
.
.
.
```

C01 1.40 © 1987 Hewlett-Packard Company

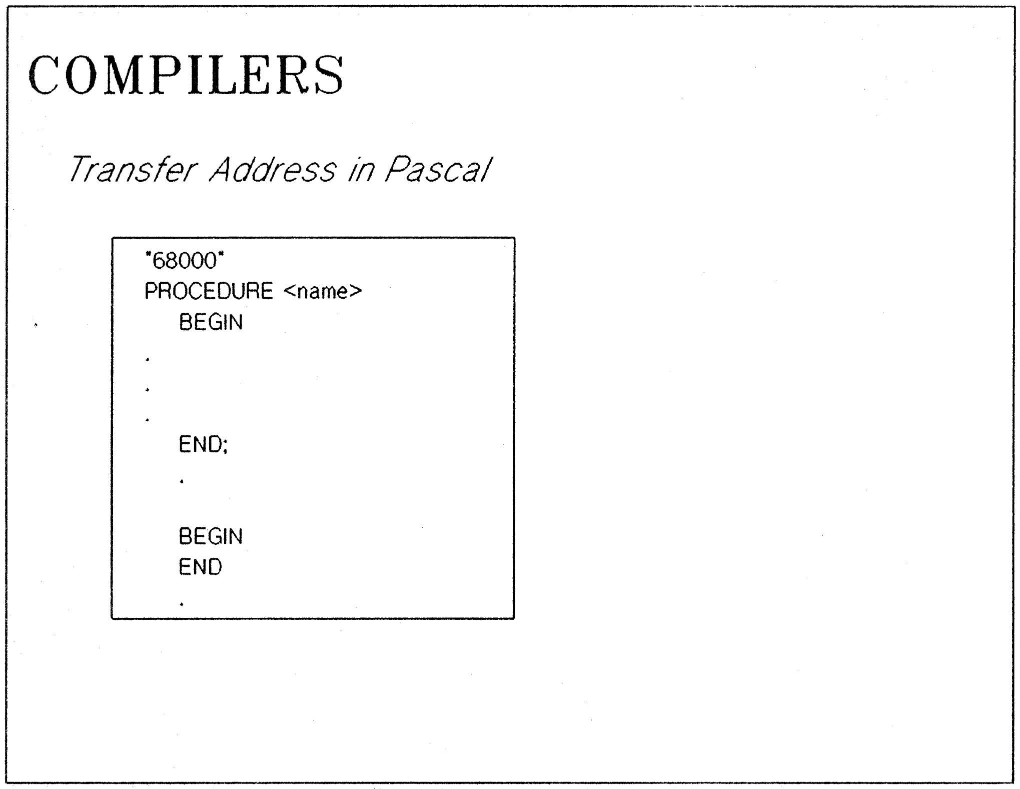
MANUAL: pages 2-9 thru 2-10 68000 C Cross Compiler

NOTES:

The transfer address in the C source code is defined by the "main ()" identifier. This lets the compiler know the start of the main program is contained in this module.

STUDENT NOTES:

☐ Compilers



COMPILERS

Transfer Address in Pascal

```
"68000"
PROCEDURE <name>
   BEGIN
.
.
.
   END;
   .

   BEGIN
   END
   .
```

C01 1.45 © 1987 Hewlett-Packard Company

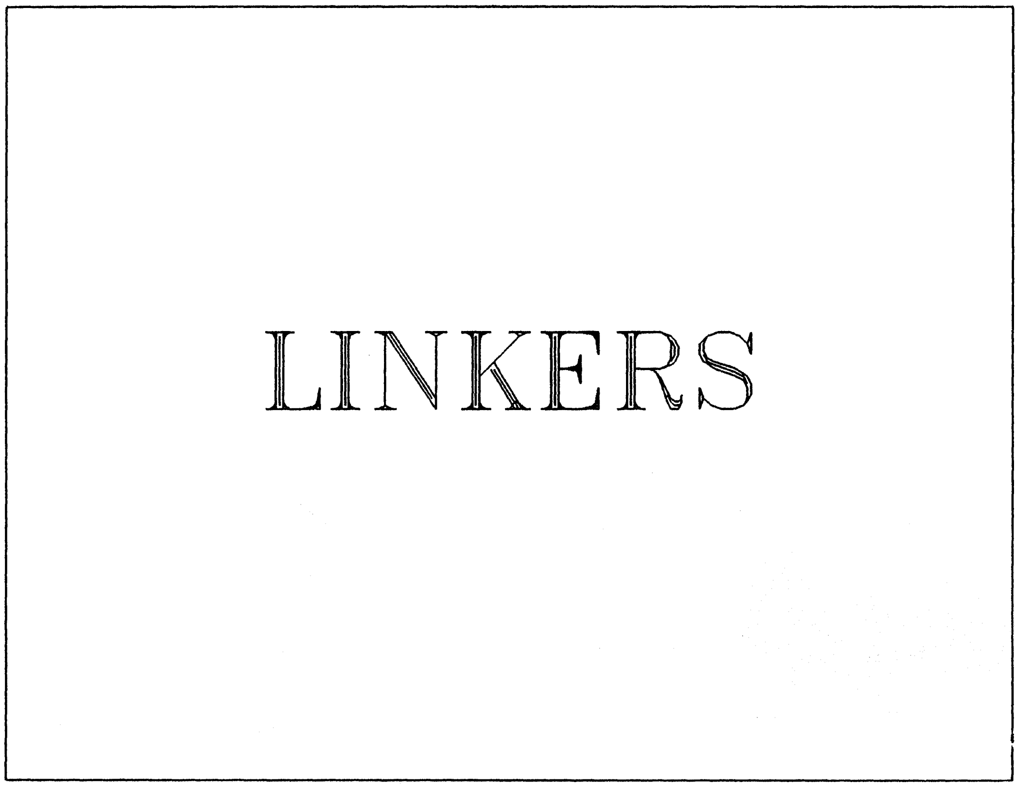
NOTES:

The transfer address in the Pascal source code is defined by the:

```
BEGIN
END
.
```

construct. This tells the compiler that the transfer address is to be defined since the main procedure is contained within this module.

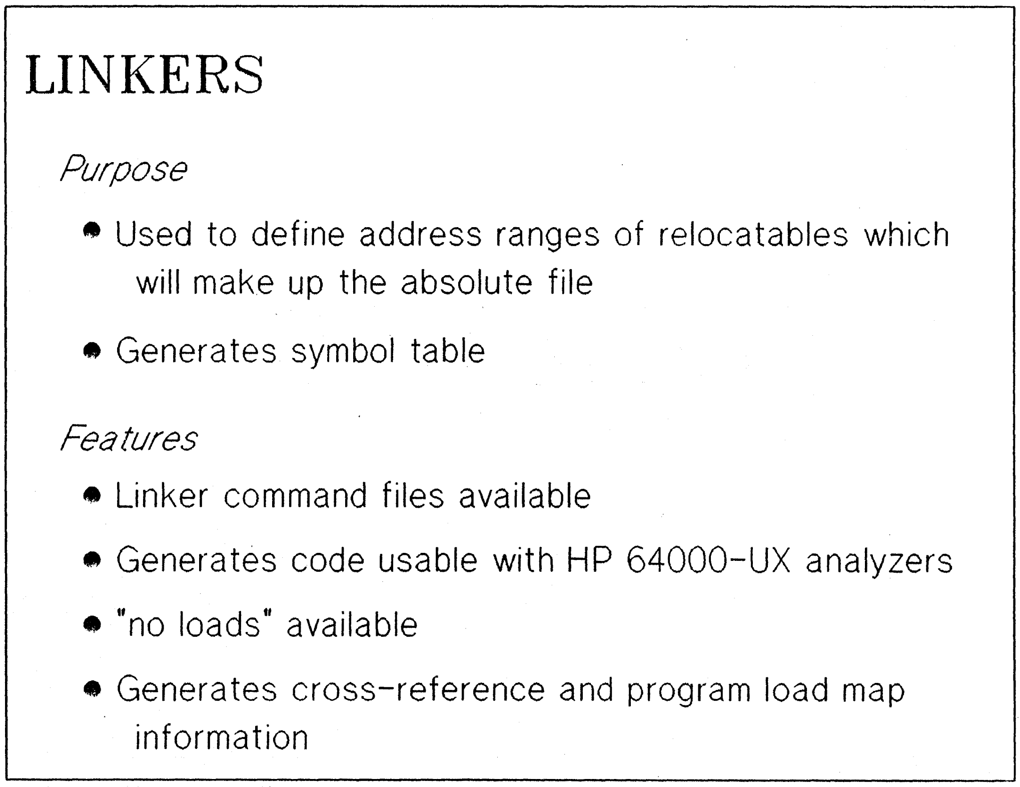
STUDENT NOTES:



LINKERS

LK1 1.05

Linkers



LINKERS

Purpose

- Used to define address ranges of relocatables which will make up the absolute file
- Generates symbol table

Features

- Linker command files available
- Generates code usable with HP 64000-UX analyzers
- "no loads" available
- Generates cross-reference and program load map information

LK1 1.10 © 1987 Hewlett-Packard Company

MANUAL: page 7-1 68000 C Cross Compiler

NOTES:

linker: used to define address ranges of the relocatables and symbolically links the relocatable modules.

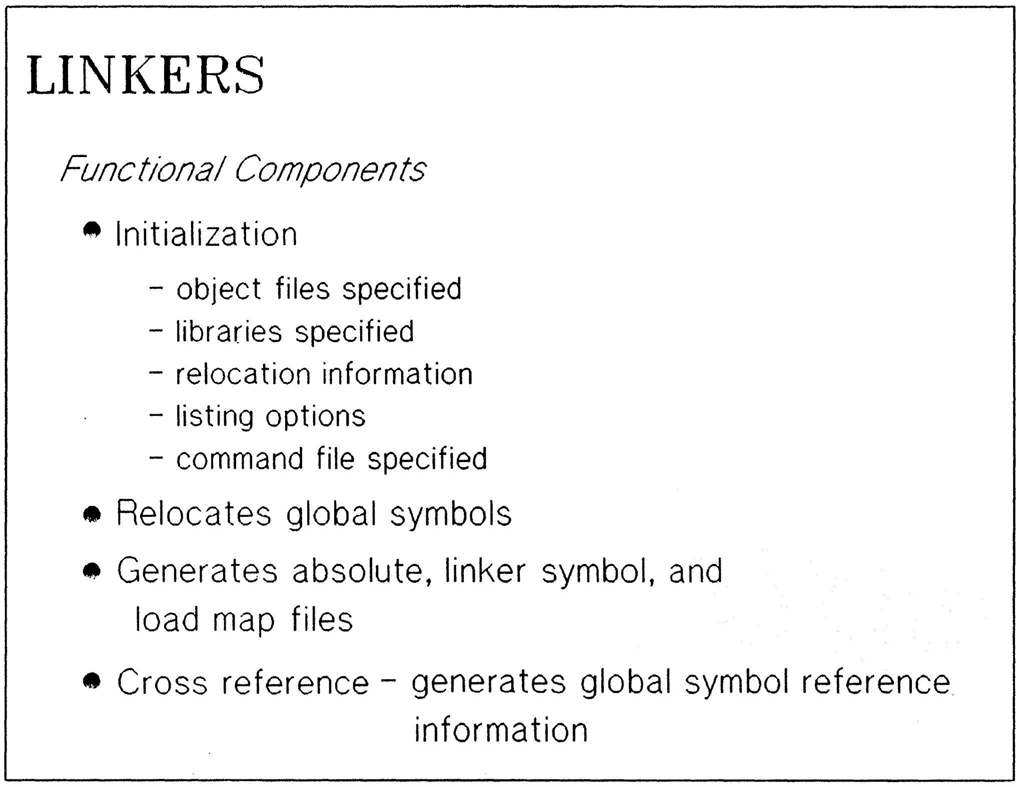
The linker symbol table assigns absolute addresses to the symbols of the relocatable modules.

no loads: files which have their symbol information loaded, but no absolute code is generated. An example would be to use no load files with emulation memory overlays (explained later).

The cross reference and program load map information are optional output of the linker and must be specified by the user.

STUDENT NOTES:

□ Linkers



LINKERS

Functional Components

- Initialization
  - object files specified
  - libraries specified
  - relocation information
  - listing options
  - command file specified
- Relocates global symbols
- Generates absolute, linker symbol, and load map files
- Cross reference – generates global symbol reference information

LK1 1.15 © 1987 Hewlett-Packard Company

MANUAL: page 7-2 68000 C Cross Compiler

NOTES:

relocation information: determines where the PROG, DATA, and COMM sections of code will be located in physical memory.

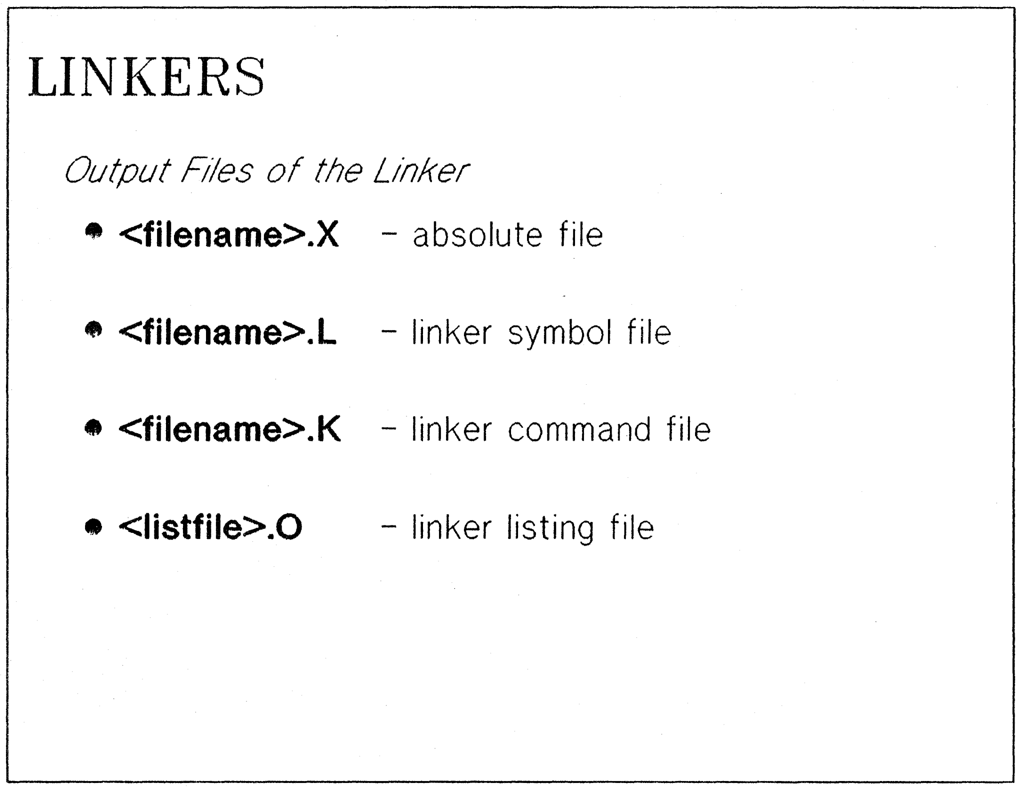
If global symbols are unknown in the object code during first pass of the compiler, the libraries specified are searched.

If memory overlays were specified by the user, they will be flagged by the linker during the second pass of the compiler.

Cross reference information includes table listings of global symbols, the relocatable which defined the global symbol, and the relocatables which reference the global symbol.

STUDENT NOTES:

Linkers



LINKERS

Output Files of the Linker

- **<filename>.X** - absolute file
- **<filename>.L** - linker symbol file
- **<filename>.K** - linker command file
- **<listfile>.O** - linker listing file

LK1 1.20 © 1987 Hewlett-Packard Company

MANUAL: pages 7-7 thru 7-10 68000 C Cross Compiler

NOTES:

filename.X: this binary file is the combination of all the relocatable files and library routines specified.

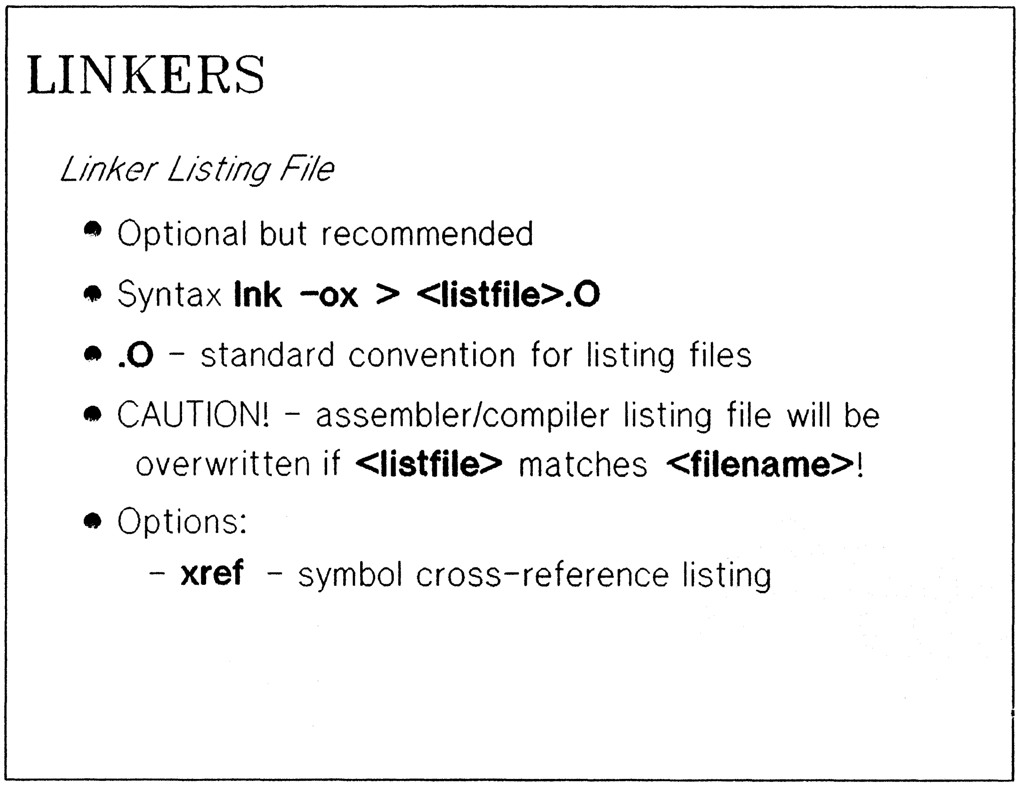
filename.L: this file contains the global symbol information from all the relocatable files and library routines used.

filename.K: this optional file is a command file containing all the information entered by the user during the linker initialization.

listfile.O: this is the optional listing file that contains the information requested by the user with the use of the linker listing options.

STUDENT NOTES:

Linkers



LINKERS

Linker Listing File

- Optional but recommended
- Syntax **lnk -ox > <listfile>.O**
- **.O** - standard convention for listing files
- CAUTION! - assembler/compiler listing file will be overwritten if **<listfile>** matches **<filename>**!
- Options:
  - **xref** - symbol cross-reference listing

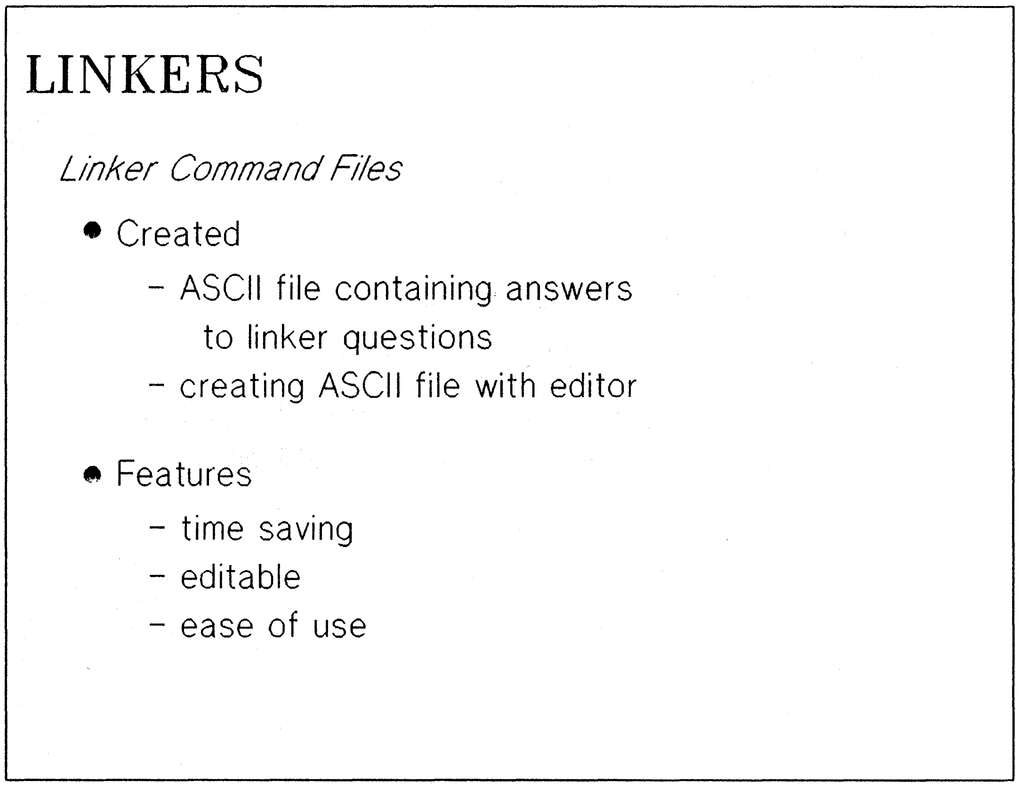
LK1 1.25 ○ 1987 Hewlett-Packard Company

NOTES:

When a linker listing file is specified as an option, a listing load map is generated similar to the one shown on page 7-7. The syntax example shown would generate linker listing file with the option "xref" turned on. An example of a cross reference table is shown on page 7-9.

CAUTION: if the filename of the linker listing matches a filename of one of the listing files of the assembler or compiler, the assembler or compiler listing file will be overwritten!

STUDENT NOTES:



LINKERS

Linker Command Files

- Created
  - ASCII file containing answers to linker questions
  - creating ASCII file with editor

- Features
  - time saving
  - editable
  - ease of use

LK1 1.30 © 1987 Hewlett-Packard Company

MANUAL: pages 7-4 thru 7-6 68000 C Cross Compiler

NOTES:

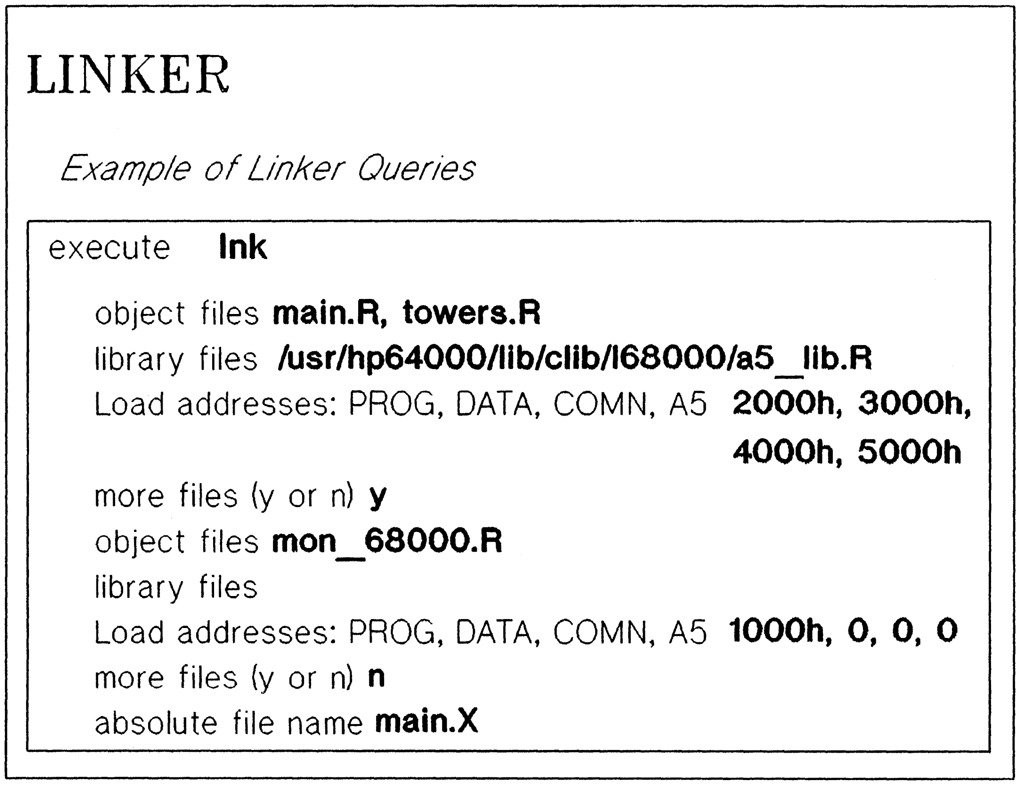
Linker command files are available so the user does not have to re-enter the same information every time the source code is changed, and the relocatables have to be re-linked.

The easiest method for creating a command file for the linker is by the interactive method. The questions asked by the linker are shown on page 7-5. The linker prompts for the relocatable and library files which will be used to create the absolute. It also prompts for the name of the absolute file to be used.

A second way to create or change a linker command file is with an ASCII editor, such as Sk. The format for the command file is given at the bottom of page 7-4.

STUDENT NOTES:

Linkers



LINKER

Example of Linker Queries

```
execute   lnk

    object files main.R, towers.R
    library files /usr/hp64000/lib/clib/l68000/a5_lib.R
    Load addresses: PROG, DATA, COMN, A5  2000h, 3000h,
                                          4000h, 5000h
    more files (y or n) y
    object files mon_68000.R
    library files
    Load addresses: PROG, DATA, COMN, A5  1000h, 0, 0, 0
    more files (y or n) n
    absolute file name main.X
```

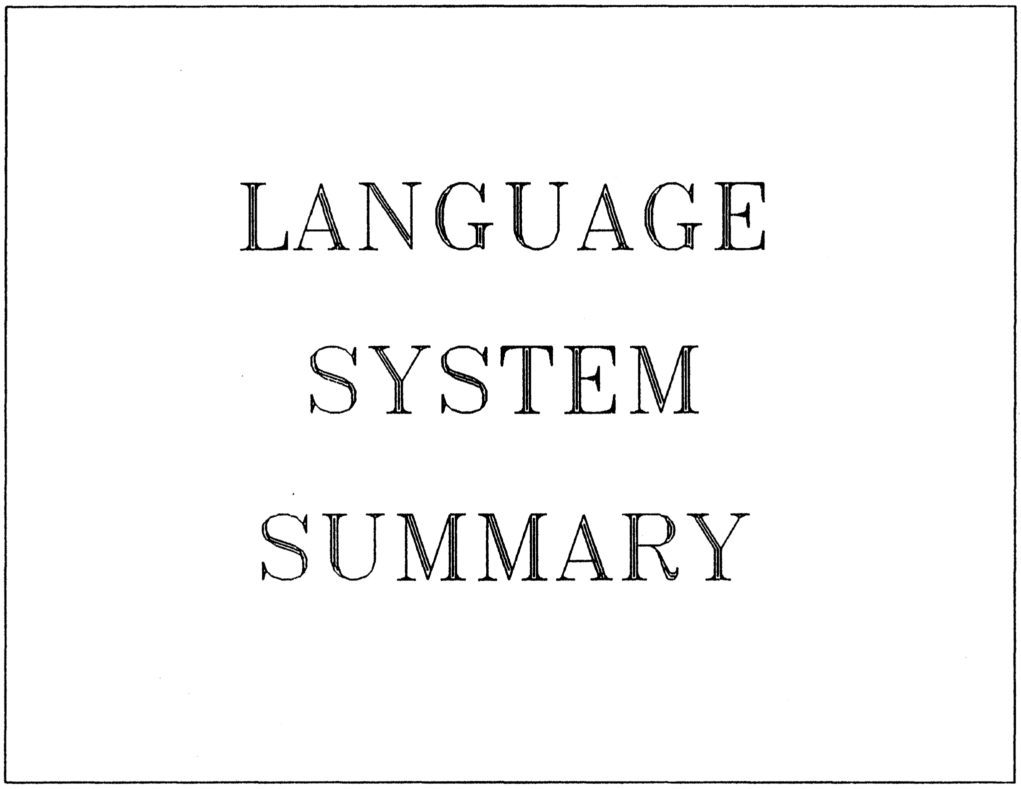
LK1 1.35 © 1987 Hewlett-Packard Company

NOTES:

NOTE: a linker command file will be created after this session with the file name "main.K".

STUDENT NOTES:

□ Language System Summary



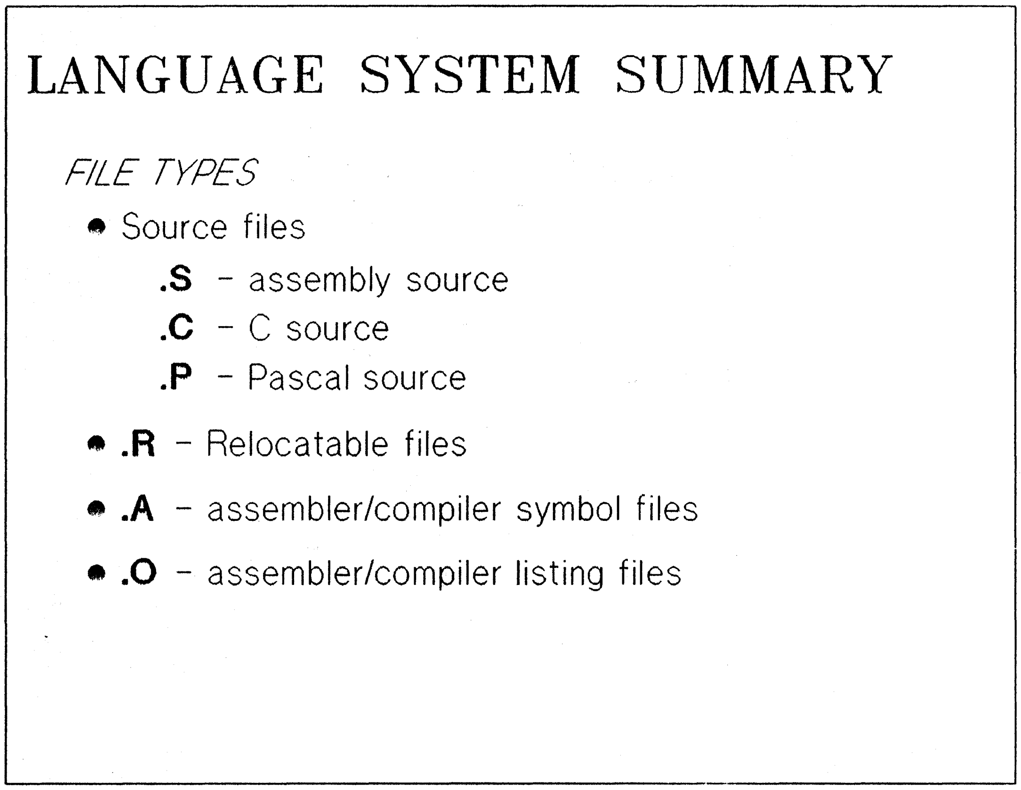
LANGUAGE

SYSTEM

SUMMARY

LS1 1.05

☐ Language System Summary



LANGUAGE SYSTEM SUMMARY

FILE TYPES

- Source files
  - **.S** - assembly source
  - **.C** - C source
  - **.P** - Pascal source
- **.R** - Relocatable files
- **.A** - assembler/compiler symbol files
- **.O** - assembler/compiler listing files

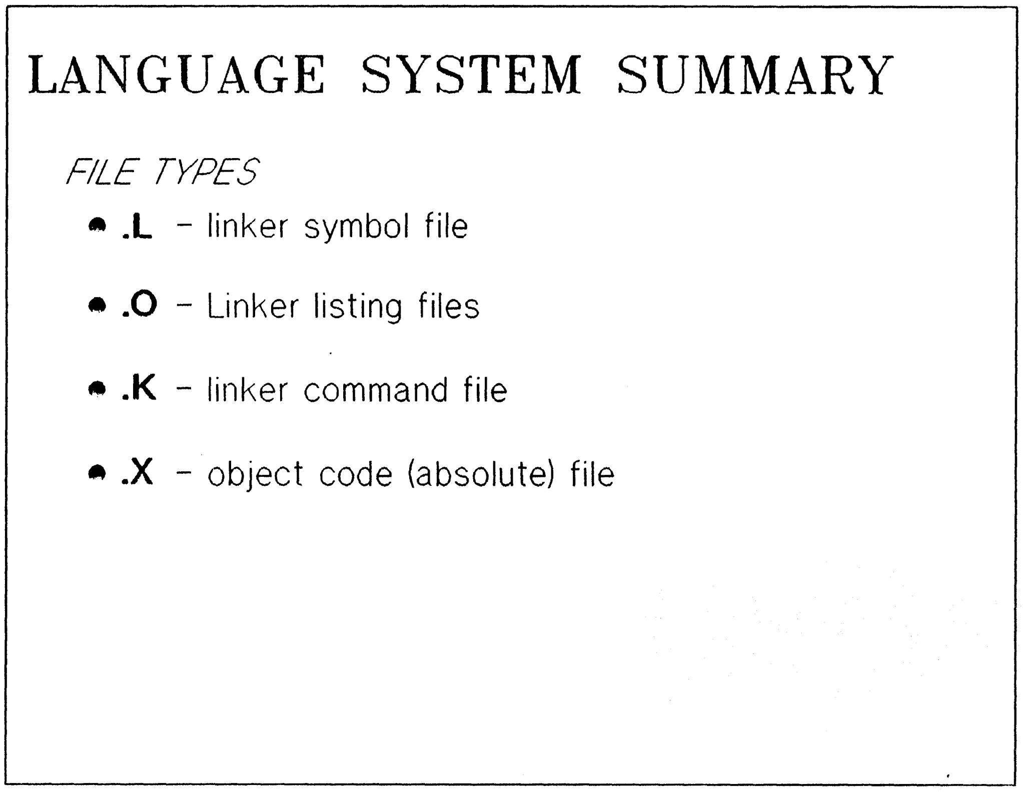
LS1 1.10

NOTES:

The filenaming conventions for the source code and assembler/compiler listing files is not required, but recommended. If this convention is followed, the user will find that the level of confusion will be less than usual.

STUDENT NOTES:

☐ Language System Summary



LANGUAGE SYSTEM SUMMARY

FILE TYPES

- **.L** - linker symbol file
- **.O** - Linker listing files
- **.K** - linker command file
- **.X** - object code (absolute) file

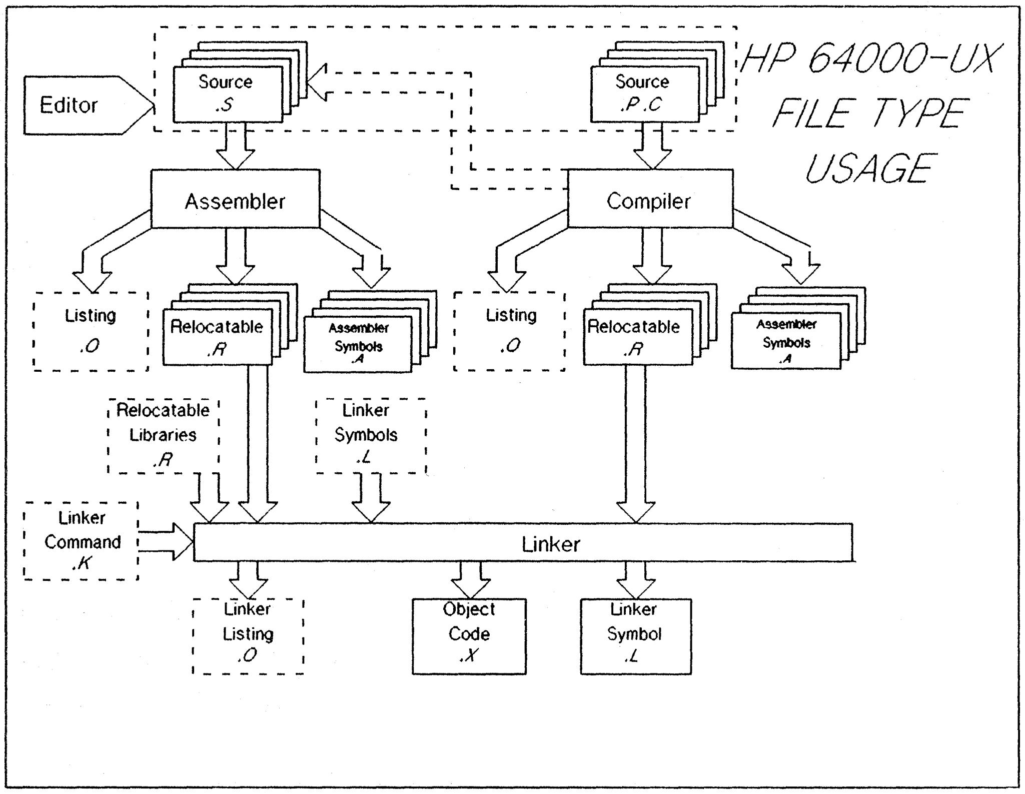
LS1 L15

NOTES:

The filenaming convention for the linker listing file is not required, but recommended. REMEMBER, do not give a linker listing file the same name as a assembler/compiler listing file, or the assembler/compiler listing file will be overwritten.

STUDENT NOTES:

☐ Language System Summary



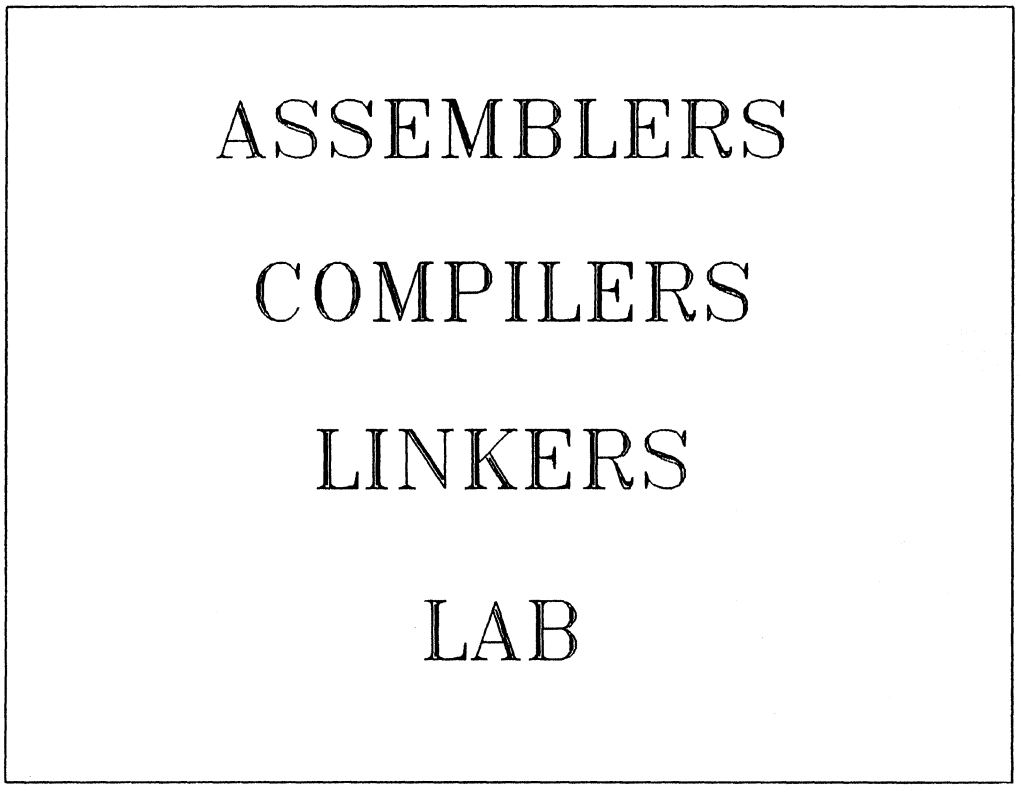
LS1 1.20 © 1987 Hewlett-Packard Company

NOTES:

The diagram above shows the complete 64000-UX Language System and the connections between its different component parts. All the files that can be generated are also shown with their respective extensions or suggested extensions.

STUDENT NOTES:

Assemblers/Linkers/Compilers Lab



ASSEMBLERS

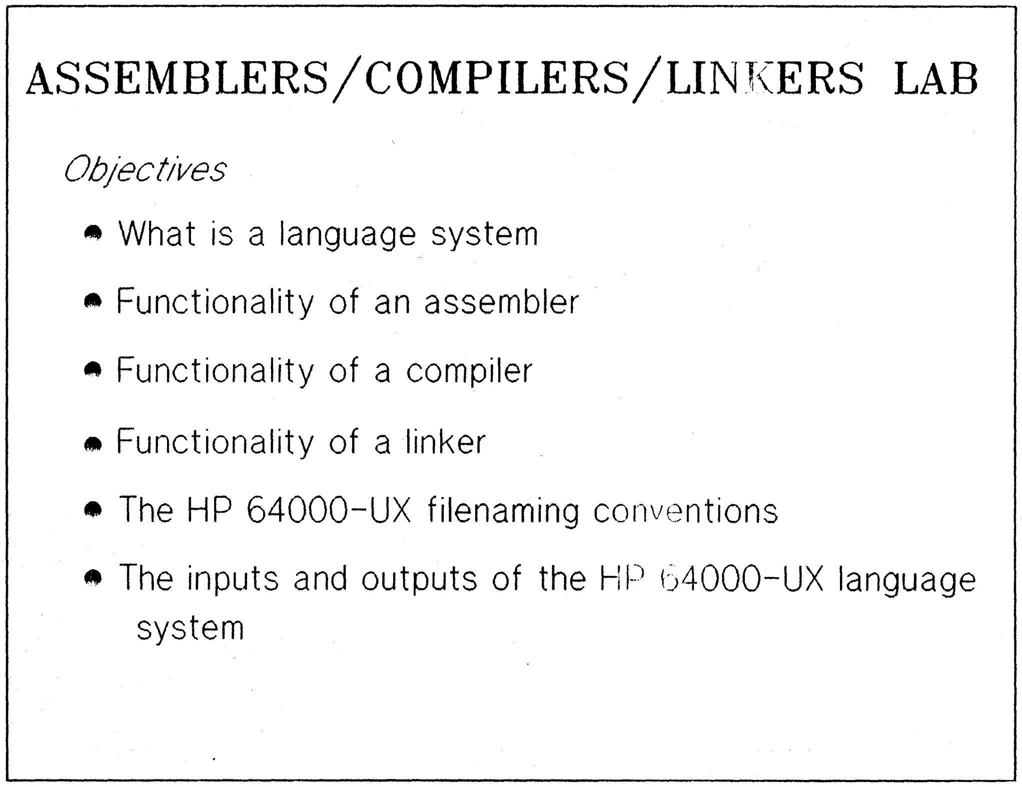
COMPILERS

LINKERS

LAB

ACI 1.05

Assemblers/Linkers/Compilers Lab



ASSEMBLERS/COMPILERS/LINKERS LAB

Objectives

- What is a language system
- Functionality of an assembler
- Functionality of a compiler
- Functionality of a linker
- The HP 64000-UX filenaming conventions
- The inputs and outputs of the HP 64000-UX language system

ACI 1.10

STUDENT NOTES:

HP 64000-UX Assemblers / Linkers / Compilers Lab

Objectives:

* understanding of a language system
* learn basic functionality of an assembler
* learn basic functionality of a compiler
* learn basic functionality of a linker
* the HP 64000-UX filenaming conventions

This lab will be broken up into four major components, the language system introduction, the assembler, the compiler and the linker.

The intent of the lab is to INTRODUCE the student to the language system tools available from Logic Systems Division which will eventually be used to generate the absolute code needed in the HP 64000-UX Development Environment.

The program to be used with this lab is a game called "Towers of Hanoi". This program is written in C for the most part, but does include a Pascal "pause" routine to show the flexibility of the language system. Another concept which will be introduced is the idea of an emulation monitor program. This program is written in 68000 assembly language and is needed to make the 68000 emulator function properly. It is user-modifiable source code and needs to be assembled and linked before using.

A brief description of each of the files used in the "Towers of Hanoi" program is shown below:

<filename>	<description> <implemented language>
mon_68000.S	the 68000 emulator monitor program (assembly)
main.C	the main routine of the Towers program (C)
pause.P	pause routine used to slow display down (Pascal)
towers.C	recursive routine for moving discs to pegs (C)
data	file for setting data parameters (C)
data_comp.C	contains data structure for discs and pegs (C)
clr_display.C	routine for clearing game display (C)
assign_discs.C	initialization routine for Hanoi (C)
gen_display.C	generate next display for game (C)
move_disc.C	procedure to remove and place discs (C)
place.C	places disc on a given peg (C)
remove.C	remove a disc from a given peg (C)
comp_hanoi	command file which assembles and links all modules
all.K	a linker command file used to link the relocatables

☐ Assemblers/Linkers/Compilers Lab

The structure of how the different routines are called is shown below:

```
                              /------- main.C ------\
                             /                       \
          gen_display.C                               towers.C
           /        \                                    |
   clr_display.C    assign_discs.C                  move_disc.C
                                                   /     |     \
                                          remove.C   place.C   pause.P
   data_comp.C ---- loaded into memory
   data        ---- included with other modules
   mon_68000.S ---- loaded into memory as emulation monitor
```

The student will need to know how to use an editor during this lab. The Sk editor is probably the best choice since it automatically supplies line numbers in the source file.

Language System

[ ] Enter pmon and access the help facilities which can give you information on the following commands:

```
edit
compile
assemble
link
log
```

Assembler

[ ] The file which will be used for this section of the lab is:
mon_68000.S

[ ] Enter the following command:
"cd" to the directory to be used for the lab; "$HOME/lang".

[ ] Enter the "pmon" software and edit the "mon_68000.S" file with the "sk" editor. Read the first 33 lines of the file to get a brief overview of an emulation monitor program.

[ ] We will now look at several features of the assembly source code:

1. The source code processor directive at the top of the file is used to define which assembler table will be used to generate the relocatable file.

2. Look at the global variable declarations on lines 36-54.

3. Examine the "ORG" assembler directive located on line 61 of the source file. The assembler will normally generate relocatable addresses for the code, but in the case of the "ORG" statement, the address reference made to these locations will be defined as absolute addresses.

4. Another important assembler directive is located on the last line of the file (664). You will notice that the "END" directive does not have an associated label. Without the label, the END directive only flags the end of the assembly routine and does not define a transfer address.

Assembling the Source

[ ] Exit the editor and save the assembly source file.

[ ] Press the "assembly" softkey from within pmon and use the HELP utility to determine what the options are available during assembly.

[ ] Assemble the "mon_68000.S" source code and generate a listfile for the output. Use the proper options so that a listing will be generated and you can monitor the assemblers progress Have the command generate a symbol cross reference as well. Name the listing file "mon_68000.O".

[ ] An error should have occurred. Use an editor to fix the bug.

[ ] Use the command recall feature of pmon (and) to recall your last assembly command and retry its execution. Verify that the assembly ran with no errors and no warnings. HINT: try the "-v" option.

[ ] Edit the listing file. NOTE: if the listing file is empty, recheck the options of the "assembly" command and fix the problem. Page down through the listing file and pay particular attention to the relative addresses assigned by the assembler. Also notice the opcode / data fields of the source file and their associated code generated by the assembler. Another point-of-interest are the line numbers which exist in the listing file.

[ ] Go to the bottom of the file. Hint: search or find the string "CROSS" which is the header for the cross reference table. Inspect this table and take note of:
- the symbols are listed in alphabetical order
- the line number where the symbol was defined is shown
- the type of symbol is given (p is for program)
- all the references made to the symbol are listed by line number

☐ Assemblers/Linkers/Compilers Lab

[ ] One of the more important symbols for the 68000 monitor is the "MONITOR_ENTRY" symbol. Take note of how many different references are made to this symbol.

[ ] Exit the editor

[ ] If you suddenly needed to check the cross reference table for a specific symbol, it would be a waste of time to enter and editor to look at the information. Instead, a much faster and more efficient method of looking at this information is with the HP-UX command "grep". Using the "grep" command determine all the line numbers which are references to the symbol "MONITOR_ENTRY".

This is a great utility for locating all the calls to a given symbol within a source file.

If you need help try: "grep MONITOR_ENTRY mon_68000.S | wc -l"

Compilers

The file which will be used for this section is:
main.C

[ ] Enter the pmon software and edit the source file main.C with the Sk editor.

[ ] The following features exist for the compiler:

1. Note the processor directive syntax on the first two lines of the file.

2. The compiler directive "ASM_FILE" is used to tell the compiler to generate an assembly code file (ASM68000).

3. Note the "extern" statements on lines 13-15. These statements define the associated variables to be global and external inside this one particular source file. Notice especially the line with "extern towers();" and the syntax of comments used by C.

4. Take note of the "main()" statement. This is the syntax for defining a transfer address in C. The transfer address will actually come from the library routine we will link later on. This library routine is discussed further in the 68000 C Cross Compiler manual on Pages 2-9 and 2-10.

☐ Assemblers/Linkers/Compilers Lab

Compile the C Source

[ ] Exit the editor

[ ] Press the "compile" softkey and use the HELP facilities to determine what options are available for the command.

[ ] Compile the file with a listing file that has the same name with a ".O" extension. Specify the options so you can monitor the progress of the compilation, a listing will be generated, and a cross reference will be generated.

[ ] A warning should have appeared. Edit the source code file and fix the bug. HINT: the error has to do with function declarations. Ask for help, if needed.

[ ] Re-compile the source code and verify it for correctness.

[ ] Edit the listing file and note the following features:
- the line numbers
- the relative addresses as generated by the compiler
- the cross reference table which shows the symbol reference information for the file.

[ ] End out of the editor and edit the file "ASM68000".
- Note the comments of the file (preceded by a ";") are the source lines of the original file, including the line numbers.
- The assembly code generated by the compiler is located directly below the source code comments.
- NOTE: this is a great feature of the compiler, but only one compiler file can have an assembly listing per directory since the "ASM68000" filename is predefined.

[ ] Exit the editor

☐ Assemblers/Linkers/Compilers Lab

Point out more features of the compilers

1. Edit the file:
   pause.P

2. Note the compiler directives on the first two lines. These tell the compiler that this file contains Pascal source code for the 68000 microprocessor.

3. The compiler directive "$EXTENSIONS ON$" allows the Pascal source to contain certain extensions to the language which are not normally available in standard Pascal.

4. A transfer address will be defined by this module with the syntax at the bottom of the source code, namely:

```
BEGIN
END.
```

Compile the Pascal source

[ ] Exit the editor

[ ] Compile the Pascal source using the options presented for the C compiler. Include the listing file.

[ ] Verify the source compiled properly.

[ ] The listing file for this compile does not differ much from the listing file of a C compiler.

Calling routines written in different source languages

[ ] This is a major benefit of the 64000-UX Language System.

[ ] Edit the file "move_disc.C" and find the call to the Pascal routine called "pause". Calling other procedures without parameters is a trivial case, but calling with parameters is beyond the scope of this course.

[ ] Exit the editor.

☐ Assemblers/Linkers/Compilers Lab

Linker

[ ] Enter pmon. Press the "link" softkey followed by the "HELP" softkey. Read the option descriptions available for the linker.

[ ] Prepare to link an example file by selecting the cross reference and output listing options of the link command. The output listing file should have the filename "main.map"

[ ] Once the command on the command line is built, press the return key to begin an interactive linking session. As an example, enter the values shown below:

```
object files            main.R
library files           /usr/hp64000/lib/clib/168000/a5_lib.R
Load addresses:         1000h,2000h,3000h,4000h
more files (y or n)     n
absolute file name      main.X
```

[ ] This command should generate any number of "undefined symbol" errors. These are due to the fact that the main routine has symbol references pointing to many of the different modules in the Towers program. The only way to fix this problem is to link in all the appropriate relocatables at one time.

[ ] Verify you are in your "$HOME/lang" directory and that the file "comp_hanoi" is also in the directory. There are more modules which need to be compiled first. Run the script "comp_hanoi" and this will compile the needed files for you. You may want to look at this file at some time because it is a good example of how to use a shell script as a command file to do work for you.

[ ] At this point, you have all the necessary modules assembled and compiled. The next step is to link the modules together to form the absolute file. You can accomplish this by using the linker command file already created for you. It is called:
    all.K
This is a good exercise in that it shows you the second way a linking session can be performed.

[ ] Run the linker and specify a linker listing file of "main.map" and use the linker command file "all.K".

☐ Assemblers/Linkers/Compilers Lab

[ ] As you noticed, the linker gave you errors telling you that multiple transfer addresses were defined. In order to give you exposure to the syntax of transfer addresses in the different languages, transfer addresses were defined in the following modules:

```
      pause.P
      main.C
```

Logically, the transfer address should reside in the "main.C" module. Edit the other module to remove the transfer address specification. Ask for help if you need to!!

HINT: the transfer address in the Pascal routine looks like:

```
    END;

    BEGIN
    END.
```

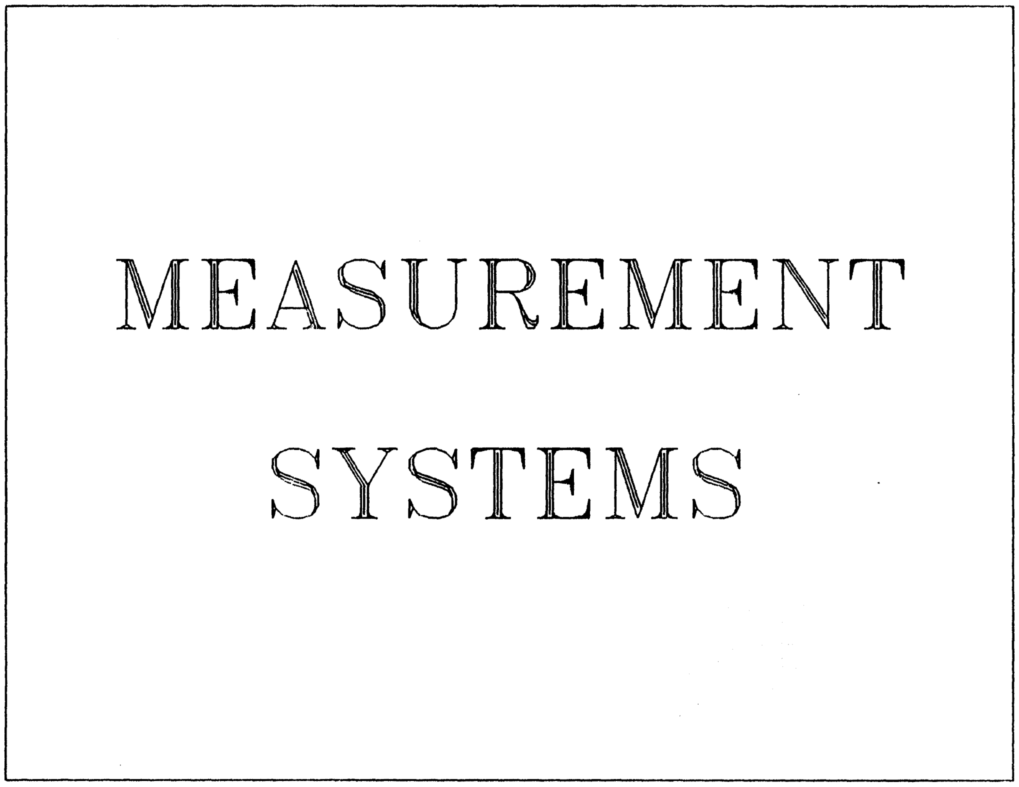
It should look like this without the transfer address:

```
    END;
    .
```

NOTE: the real transfer address will be defined in the library routine mention above. The 64000-UX C compilers know to include this routine as the transfer address so that proper functioning of the 64000-UX emulators can be preserved.

Refer to pages 2-9 and 2-10 of the 68000 C Cross Compiler manual for more thorough coverage of this topic!

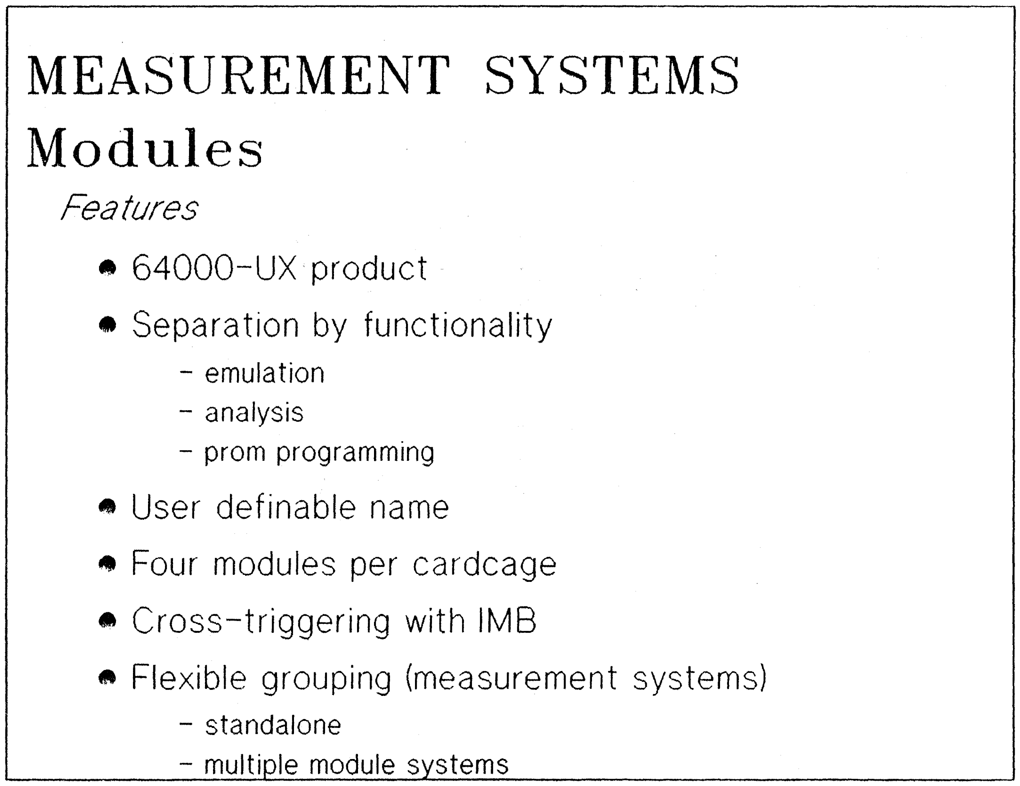
[ ] Don't forget, the modified file has to be re-assembled and re-compiled. Now try to re-link the modules using the "all.K" linker command file. If all is well, you have done it!!



MEASUREMENT SYSTEMS

MS1 1.05

Measurement Systems



MEASUREMENT SYSTEMS

Modules

Features

- 64000-UX product
- Separation by functionality
  - emulation
  - analysis
  - prom programming
- User definable name
- Four modules per cardcage
- Cross-triggering with IMB
- Flexible grouping (measurement systems)
  - standalone
  - multiple module systems

MS1 1.10 © 1987 Hewlett-Packard Company

MANUAL: pages 1-1 thru 1-3 HP 64000-UX Measurement Systems

NOTES:

module: a single 64000-UX product installed in the cardcage.

Example modules:

64243S	68000 emulation subsystem
64310A	software performance analyzer
64620A	state analyzer
64610S	timing analyzer
64501A	prom programmer

A single module can consist of several boards, i.e. the 68000 emulation subsystem you will use in lab consists of the following hardware components:

64243	68000 emulation control card
64243	68000 emulation pod
64302A	Emulation bus analyzer (internal analyzer)
64155B	Emulation memory controller with static RAM

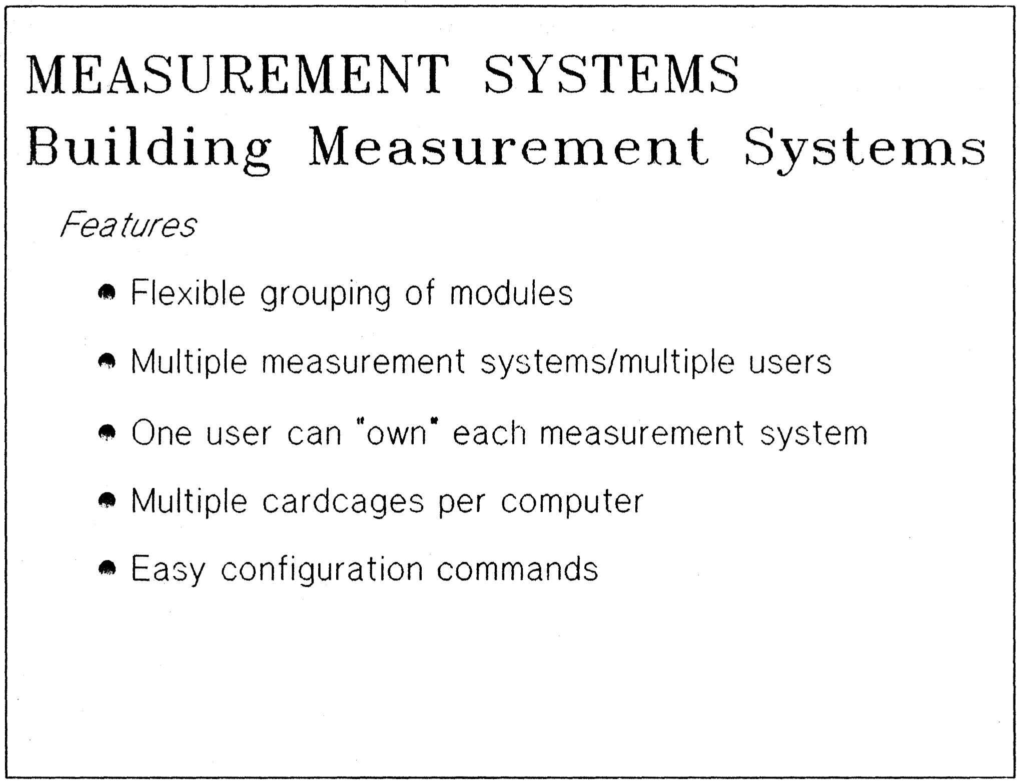
(or equivalent memory subsystem: 64155A and 64162A)

☐ Measurement Systems

IMB (inter-module bus): the 64000-UX analyzer communication bus. Allows features such as driving triggers, receiving triggers, and analyzer enable signals.

An example of IMB connections is shown on page 1-5 of the 64000-UX Measurement System manual.

STUDENT NOTES:



MEASUREMENT SYSTEMS

Building Measurement Systems

Features

- Flexible grouping of modules
- Multiple measurement systems/multiple users
- One user can "own" each measurement system
- Multiple cardcages per computer
- Easy configuration commands

MS1 1.15 © 1987 Hewlett-Packard Company

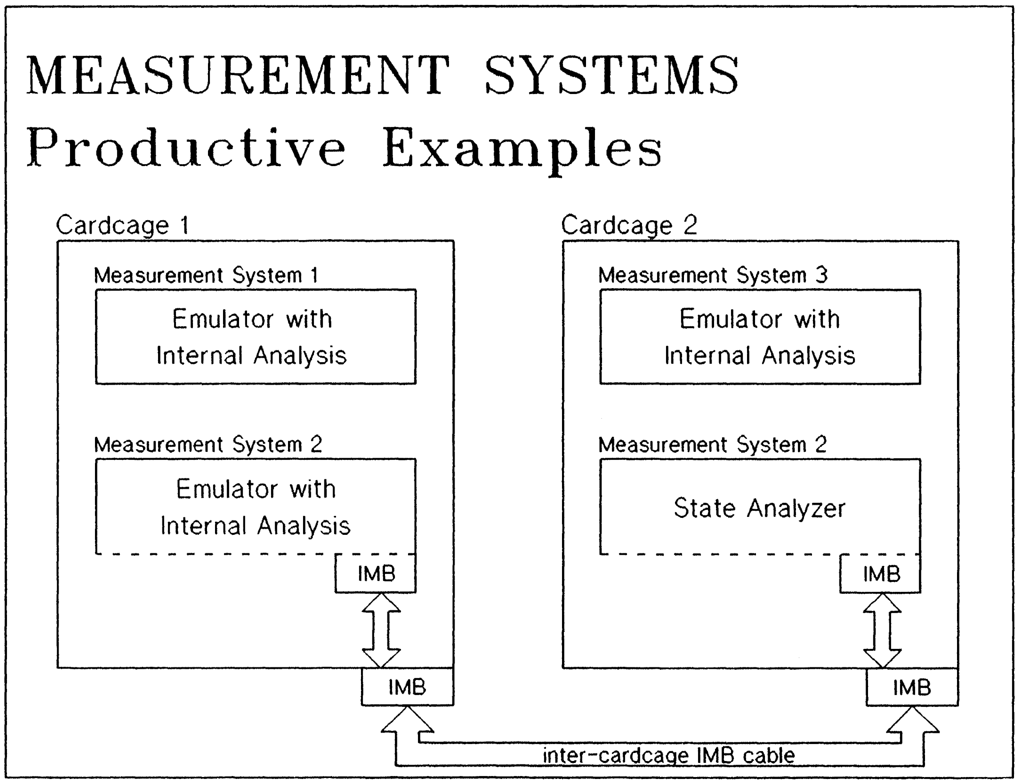
NOTES:

measurement system: one or more modules connected together with an IMB cable for the purpose of coordinating measurements between the modules.

HP 64000-UX limitations:
- up to six modules can be in one measurement system
- only one measurement system can cross between cardcages due to IMB cabling constraints
- up to four measurement systems per cardcage
- maximum of four measurement systems running simultaneously

STUDENT NOTES:

Measurement Systems



MEASUREMENT SYSTEMS
Productive Examples

MSI 1.20 © 1987 Hewlett-Packard Company

MANUAL: page 1-3 Measurement Systems Manual

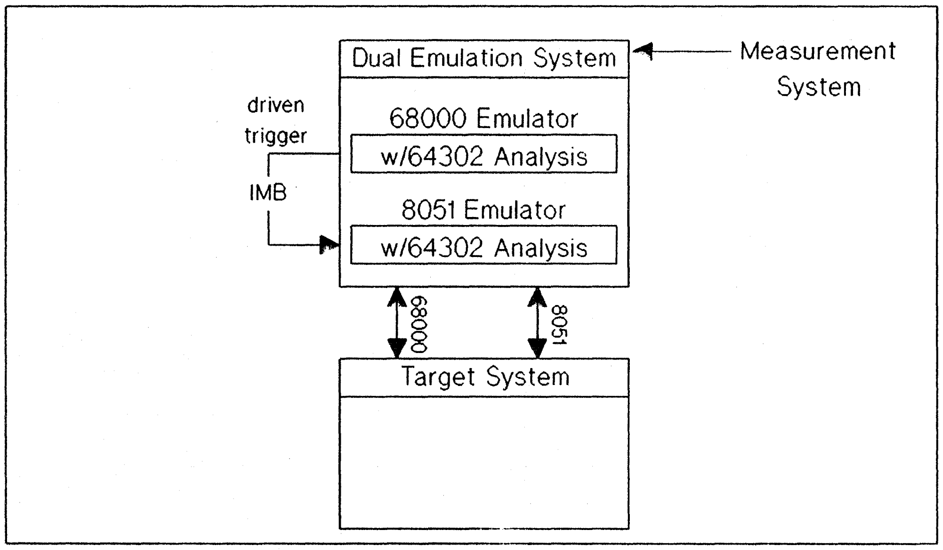
NOTES:

The above slide shows a possible two cardcage configuration which contains three emulation systems. One of these systems also utilizes the state analyzer and the 64000-UX IMB.

STUDENT NOTES:

☐ Measurement Systems

MEASUREMENT SYSTEMS
Productive Configurations



MSI 1.25 © 1987 Hewlett-Packard Company

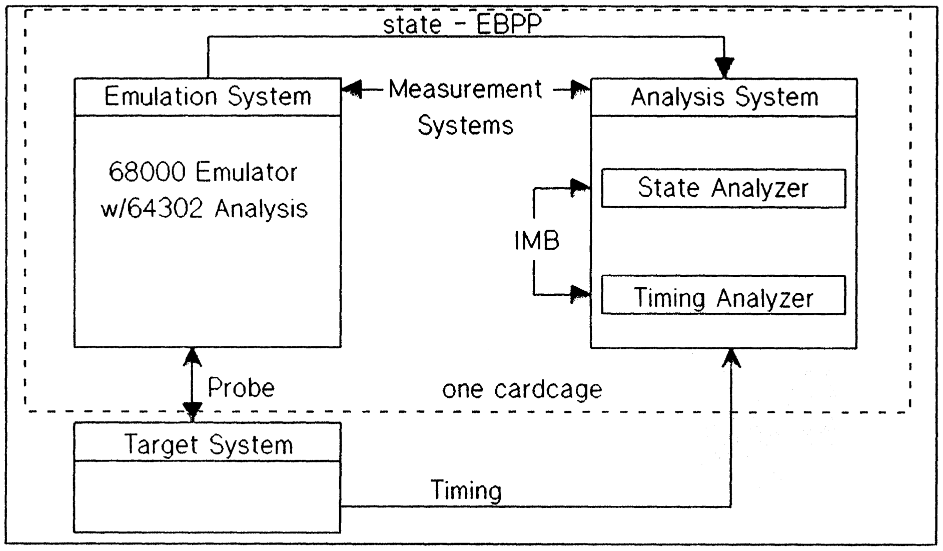
NOTES:

The above dual emulation system shows one possible interaction between a 68000 emulator and an 8051 microcontroller. The 68000 can be used to detect a trigger condition and to activate the 8051 for further software debugging.

STUDENT NOTES:

☐ Measurement Systems

MEASUREMENT SYSTEMS Productive Configurations



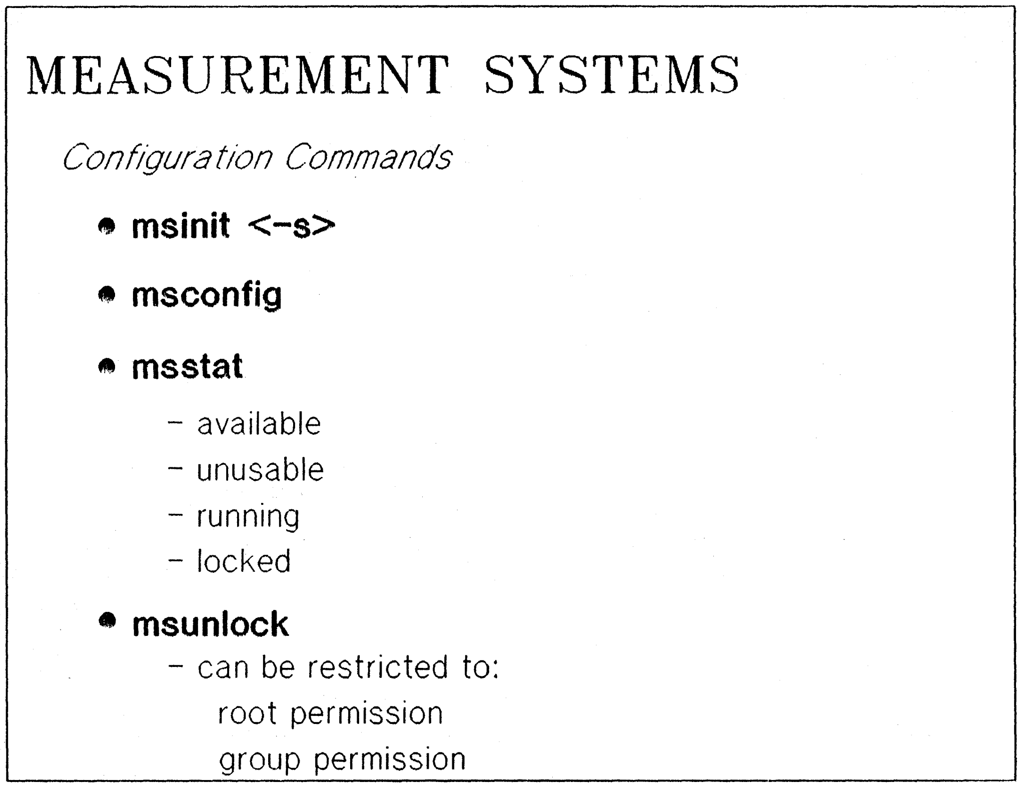
MSI 1.30

NOTES:

The above slide shows the flexibility and power of the HP 64000-UX Development System. This configuration combines the 68000 emulator with the power of the state analyzer and timing analyzer as an efficient hardware and software debugging tool.

STUDENT NOTES:

Measurement Systems



MEASUREMENT SYSTEMS

Configuration Commands

- **msinit <-s>**
- **msconfig**
- **msstat**
  - available
  - unusable
  - running
  - locked
- **msunlock**
  - can be restricted to:
    root permission
    group permission

MS1 1.35 © 1987 Hewlett-Packard Company

MANUAL: pages 2-1 thru 2-9 HP 64000-UX Measurement Systems manual

NOTES:

MANUAL: pages 2-1 thru 2-3 HP 64000-UX Measurement Systems manual

msinit: initializes the cardcages and all modules contained in the cardcages.

The "-s" option is used whenever a cardcage is added or deleted from the 9000 series 300 computer. It should also be used when changes are made to the IMB cabling. A full list of these situations is on page 2-1 of the manual.

MANUAL: pages 2-4 thru 2-7 HP 64000-UX Measurement Systems manual

msconfig: a softkey driven interface which allows the grouping of available modules into measurement systems.

The "msconfig" command brings up its own set of softkeys. These commands are summarized on page 2-7 of the manual.

☐ Measurement Systems

MANUAL: pages 2-8 thru 2-9 HP 64000-UX Measurement Systems manual

msstat: displays the status of the defined measurement systems.

The following are possible states of measurement systems:

available	- system is initialized and ready for use
running	- a user is running the system
locked	- a user has temporarily stopped using the system
unusable	- hardware is improperly setup or the power is off

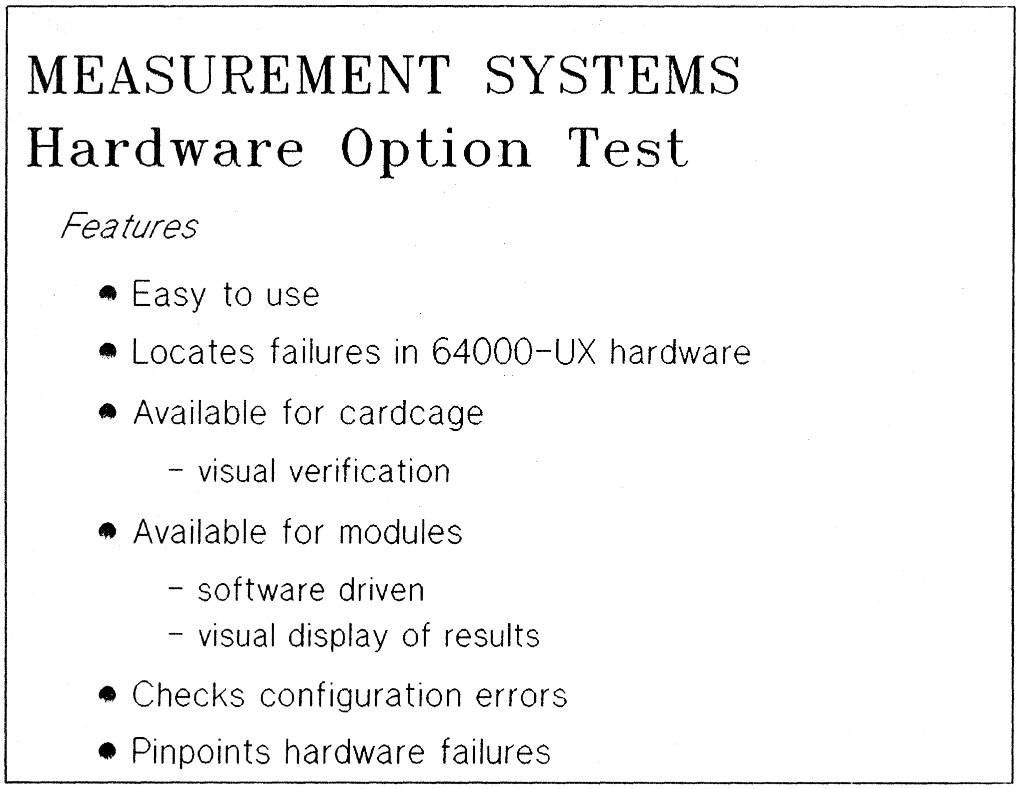
MANUAL: pages 3-6 HP 64000-UX Measurement Systems manual

msunlock: used to "free up" measurement systems locked by other users.

The "msunlock" command can be controlled by modifying the execution permissions of the command itself
The access can be restricted to:

"root" permission
"group" permission
"all others" permission

STUDENT NOTES:



MEASUREMENT SYSTEMS
Hardware Option Test

Features

- Easy to use
- Locates failures in 64000-UX hardware
- Available for cardcage
  - visual verification
- Available for modules
  - software driven
  - visual display of results
- Checks configuration errors
- Pinpoints hardware failures

MS1 1.40 ○ 1987 Hewlett-Packard Company

MANUAL: pages 9-1 thru 9-6 HP 64000-UX User's Guide

NOTES:

option test: is a program which tests 64000-UX hardware for failures.

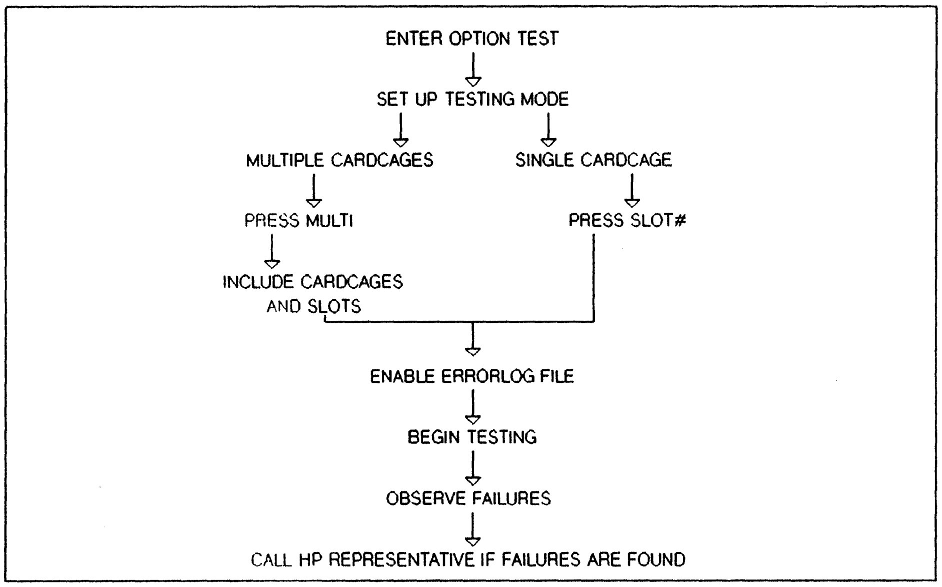
Option test should be ideally performed every time a module is installed or reinstalled in the cardcage. It should also be run if the user suspects that the hardware is the cause of operational problems.

Option test has the ability to pinpoint a hardware failure to a specific cardcage and to a specific board in that cardcage.

STUDENT NOTES:

☐ Measurement Systems

MEASUREMENT SYSTEMS
Hardware Option Test



MS1 1.45 © 1987 Hewlett-Packard Company

MANUAL: pages 9-2 thru 9-5 HP 64000-UX User's Guide

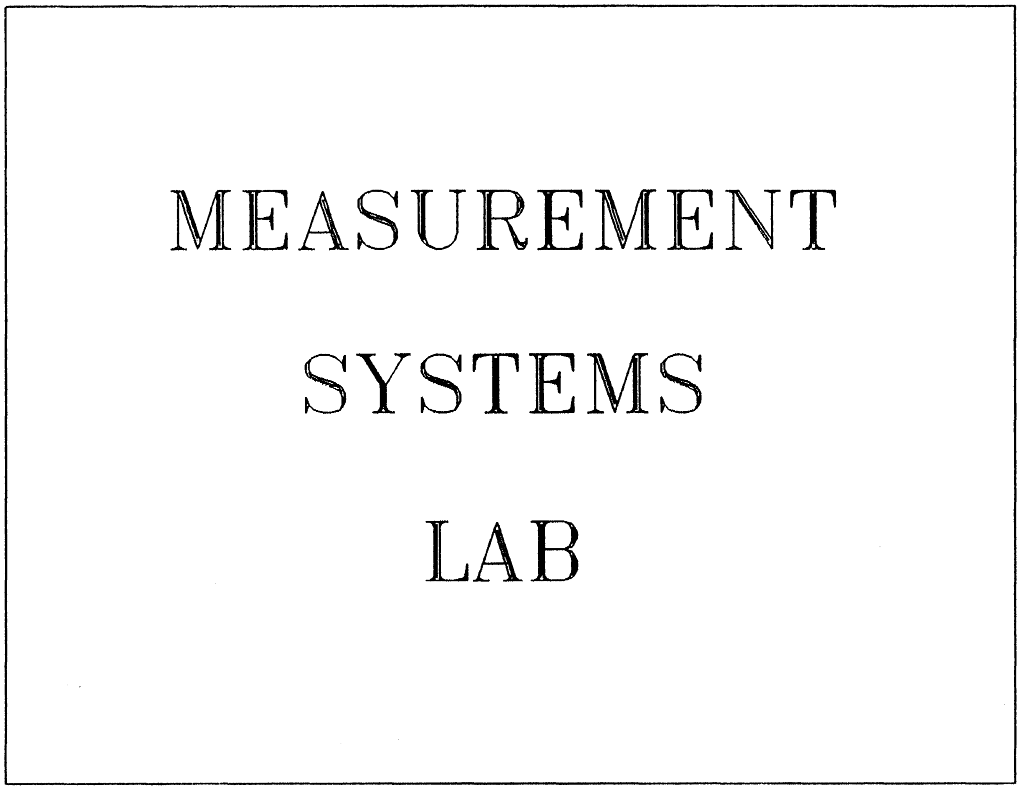
NOTES:

The above flow diagram shows the steps involved in testing 64000-UX hardware.

The methods of accessing the option test and using the option test should be discussed in the lecture (User's Guide - Chapter 9). You will get to perform the option test on the emulation hardware in lab.

Descriptions of the "Options Test Softkeys" are shown on page 9-5 of the User's Guide.

STUDENT NOTES:

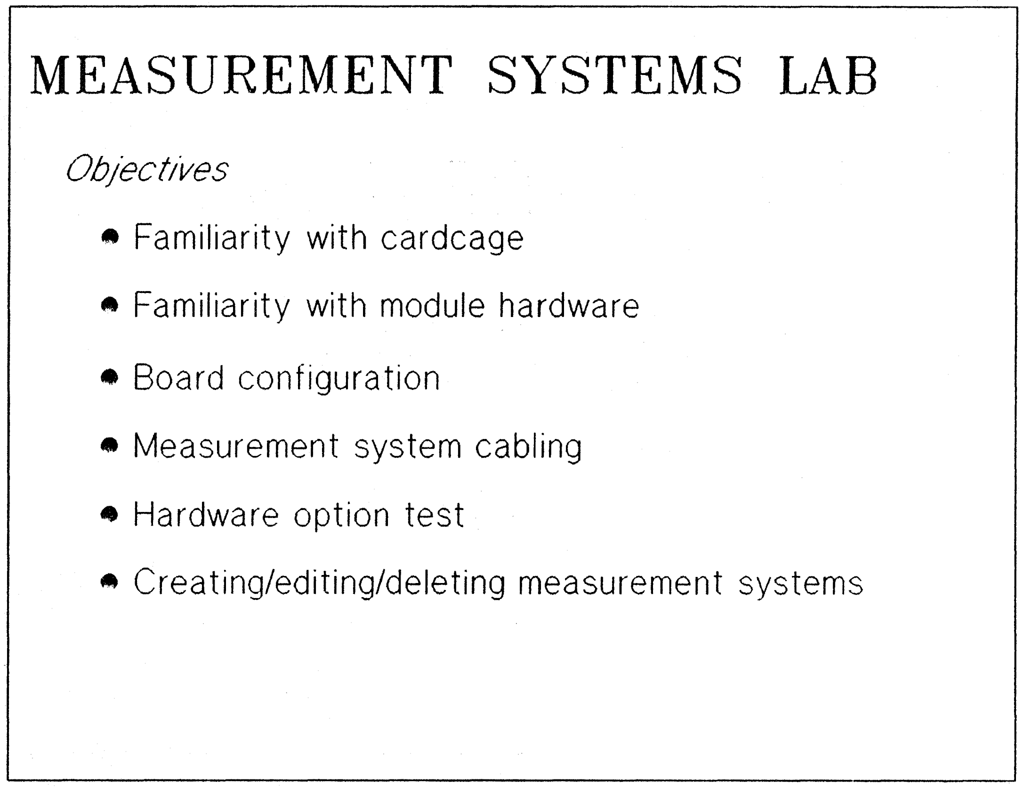


MEASUREMENT

SYSTEMS

LAB

MS1 1.50



MEASUREMENT SYSTEMS LAB

Objectives

- Familiarity with cardcage
- Familiarity with module hardware
- Board configuration
- Measurement system cabling
- Hardware option test
- Creating/editing/deleting measurement systems

MSI 1.55

STUDENT NOTES:

HP 64000-UX Measurement Systems Lab

The HP 64000-UX Measurement Systems manual is an excellent source of information for this lab.

Objectives:

* familiarity with cardcage
* familiarity with module hardware
* board configuration
* measurement system cabling
* hardware option test
* creating / editing / deleting measurement systems

The beginning of this lab will be a short explanation of the hardware associated with a measurement system, specifically the 68000 emulation module with internal analysis.

The lab will be broken up into three basic parts: hardware configuration, hardware option test, and measurement systems

Hardware Configuration

[ ] A look at the 64120 instrumentation cardcage
- rear panel
  - power connector
  - voltage selector
  - fuse
  - power switch
  - HP-IB
    - connection to the 9000 series 300
    - address and XFT switches
  - external IMB extender connector
  - BNC ports
- inside of cardcage
  - motherboard
  - IMB connector for bus
  - cabling grounds and restraints
- front panel
  - power indicator
  - two self-test "passed" indicators

□ Measurement Systems Lab

Installing the Hardware

```
[ ]  Installing the emulation hardware
          - inspection of the module boards and functionality
               - method of board layout (bus configuration)
               - emulation control board
               - emulation pod
               - memory system (64155B or equivalent)
                    - contains both the memory controller and memory
               - internal analyzer (64302)
               - different buses: emulation bus, analysis bus, memory bus
          - installation of boards
               - memory system
               - analyzer
               - emulation control card and its connection to the pod
                    - remove probe protector
               - install memory bus cable
               - install analysis bus cable
               - show possible installation of IMB cables
               - proper emulation pod cable routing (include restraint)
          - verify proper hardware configuration
               - boards seated properly?
               - cables seated and routed properly?
```

Running Option Test

```
[ ]  HP 64120 Option Test
          - secure power cord
          - switch power ON
          - verify cardcage option test
               - AC power indicator (bottom LED)
               - two self test lights (top two LEDs)

[ ]  Emulation Option Test
          - The option test software is accessed through the pmon
            interface.  Enter pmon and press the "opt_test" softkey.
          - Then read the available help messages for this level of
            softkeys.  Start the options test without using any of the
            command options available.
          - Use the "help" softkey at this level to help you select the
            68000 emulator for the option test.  If a message appears that
            the hardware is in use, ask for help from the instructor.

          - When no one else is accessing the hardware run the option
            test to cycle through the board set at least five times to
            verify proper hardware operation.  Enable an error log file
            in the directory "/tmp" when running option test.

          - End the hardware option test only if no errors in the hardware
            configuration were found.
```

☐ Measurement Systems Lab

Building and Maintaining Measurement Systems

[ ] Running "msinit"
- Enter the measurement system interface software by using the "MEAS_SYS" softkey in pmon.
- Use the on-line help facilities to determine the softkey functions.
- Initialize the cardcage and module hardware as if you have just connected the cardcage to the 9000 series 300.
- Verify no errors were found with the configuration
- Verify displayed address matches the address set on cardcage with switches

CAUTION: DO NOT run msinit when another user is running the hardware option test.

NOTE: When naming measurement systems, be very careful your name does not match the name given by another user on the same computer. Always use a unique name, such as part of your login name.

[ ] Running "msconfig"
- execute the msconfig command
- make a measurement system
  - type the command "help" and select the softkey associated with the msconfig commands. Determine the function of each of the msconfig commands.
  - name the measurement system "<username>", where <username> is your login name.
  - name the 68000 emulation module "M68000"
  - include only the 68000 module
- edit the measurement system
  - delete the 68000 module
  - remove the measurement system "<username>"
- recreate a new measurement system and name it "<username>"
  - add in the 68000 module with name "hanoi"

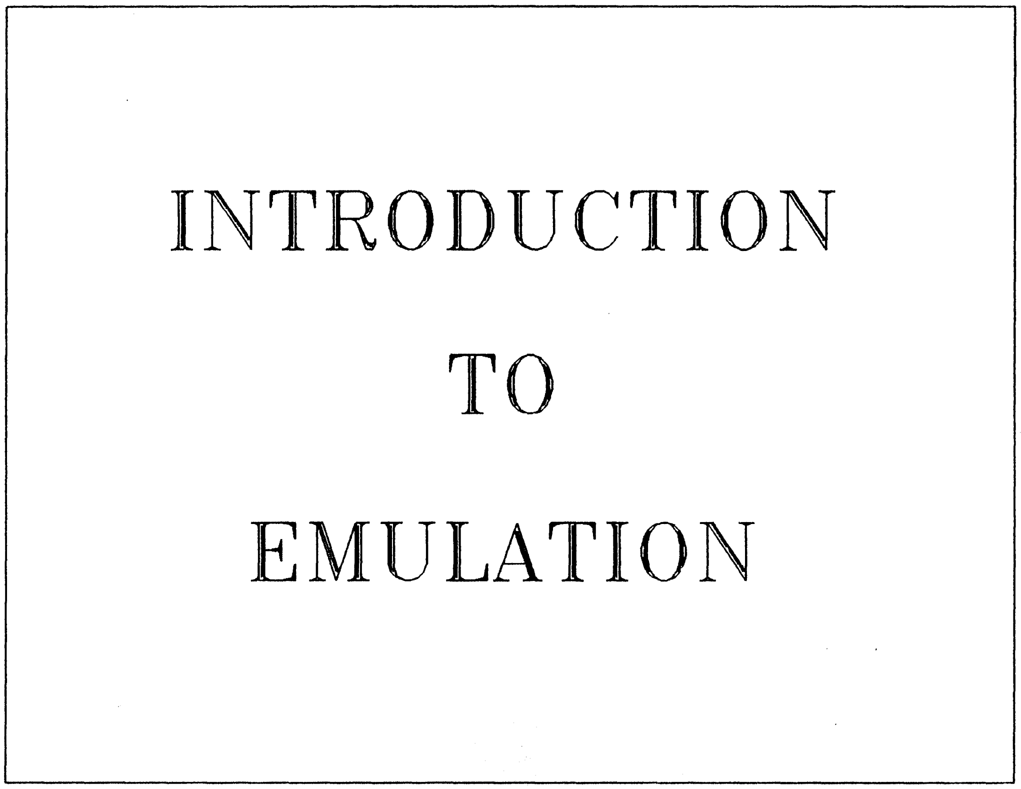
[ ] Running "msstat"
- execute the msstat command
- verify your measurement system is named properly
- verify that your measurement system is "available"

- if your measurement system is:
  unusable
  locked
  running
  try to troubleshoot why and remedy the situation

☐ Measurement Systems Lab

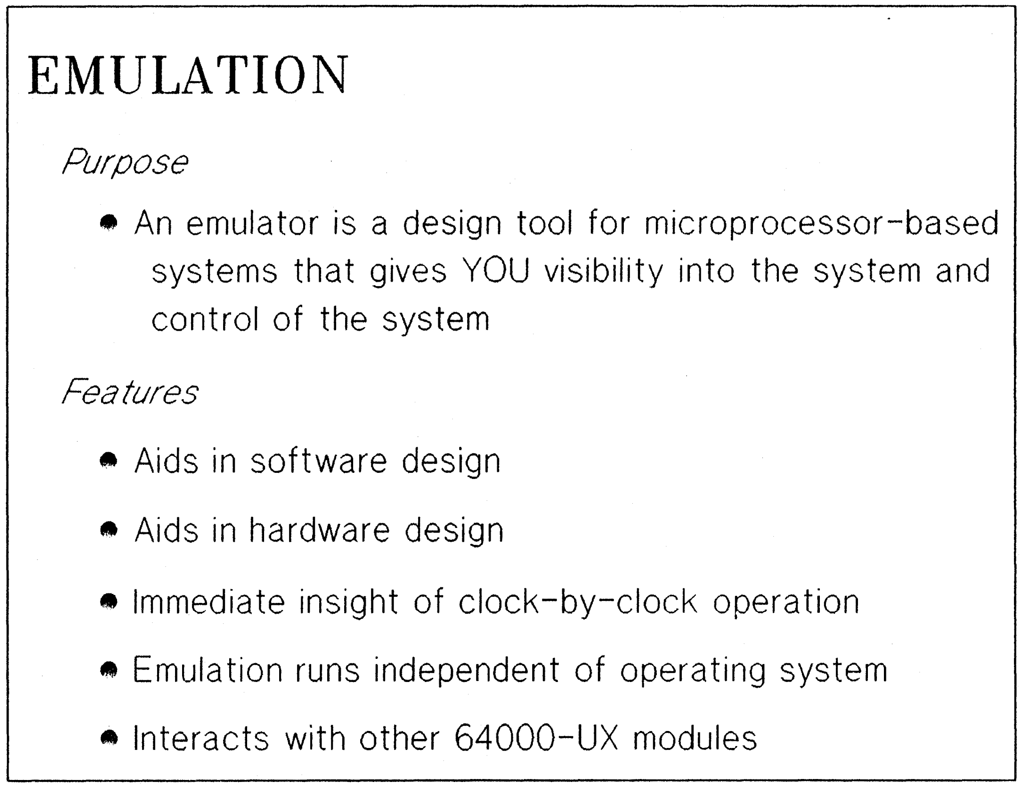
[ ] Running "msunlock"

- enter MEAS_SYS software
- press the "<username>" softkey and then the "hanoi" softkey and press return
- enter "msstat" as an HP-UX command and verify that the measurement system hanoi is running. Press return. HINT: use a "!" in front of the command to make it an HP-UX command.
- when the next level of softkeys appears, press the "end" softkey and immediately press return.
- The above sequence of commands will simulate a locked measurement system.
- use msstat to determine the current status of your measurement system
- the measurement system should now be "locked"
- use the help facilities and msunlock to make the measurement system available again.
- verify that what you tried worked



INTRODUCTION TO EMULATION

EM1 1.05



EMULATION

Purpose

- An emulator is a design tool for microprocessor-based systems that gives YOU visibility into the system and control of the system

Features

- Aids in software design
- Aids in hardware design
- Immediate insight of clock-by-clock operation
- Emulation runs independent of operating system
- Interacts with other 64000-UX modules

EM1 1.10

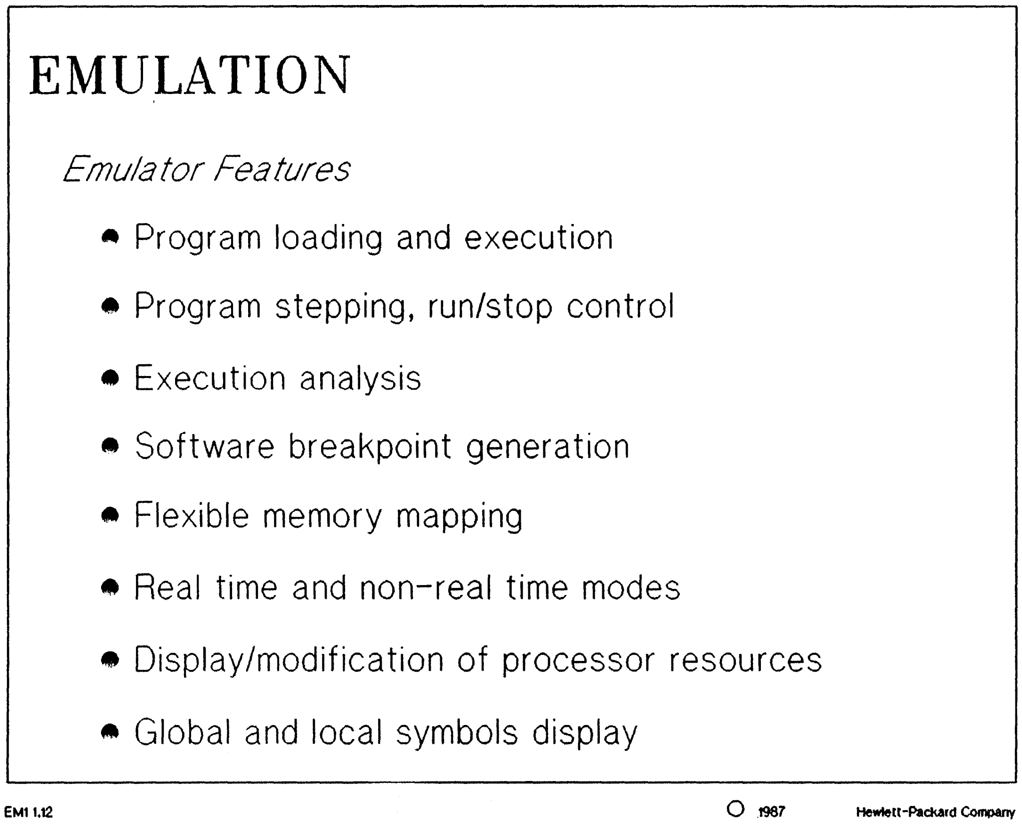
NOTES:

The emulator can gain insight into the processor's clock-by-clock operation by utilizing the analysis and monitor capabilities of the emulator.

Once the operating system (HP-UX) has loaded the appropriate 64000-UX software, the development system no longer requires direct intervention from the operating system.

STUDENT NOTES:

Introduction to Emulation



EMULATION

Emulator Features

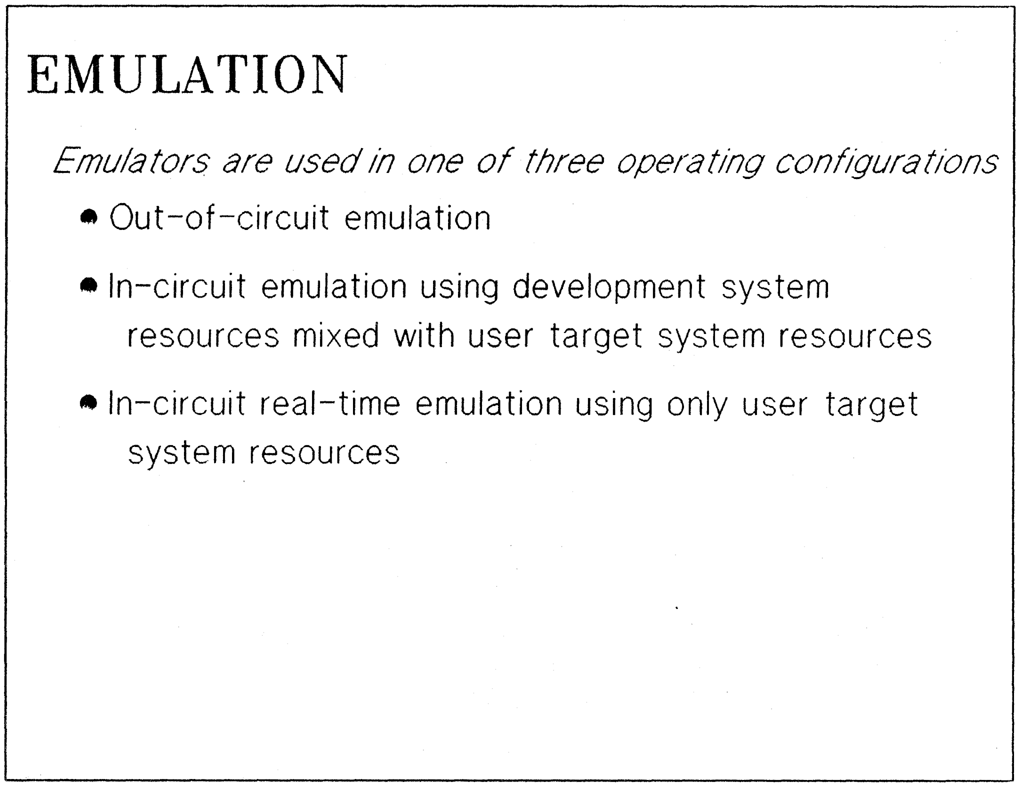
- Program loading and execution
- Program stepping, run/stop control
- Execution analysis
- Software breakpoint generation
- Flexible memory mapping
- Real time and non-real time modes
- Display/modification of processor resources
- Global and local symbols display

EM1 1.12 © 1987 Hewlett-Packard Company

NOTES:

The software breakpoint generation feature is available in the 68000 emulator, but not available in all the 64000-UX emulators.

STUDENT NOTES:



EMULATION

Emulators are used in one of three operating configurations

- Out-of-circuit emulation
- In-circuit emulation using development system resources mixed with user target system resources
- In-circuit real-time emulation using only user target system resources

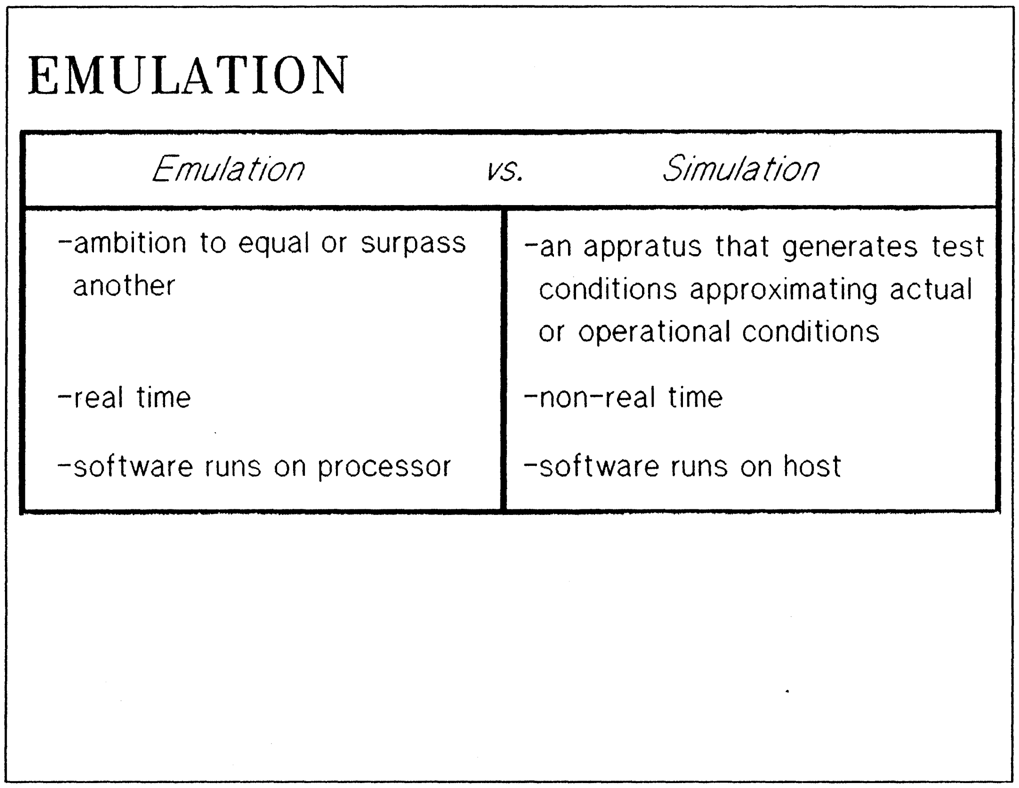
EM1 1.15 © 1987 Hewlett-Packard Company

NOTES:

This course is designed with out-of-circuit emulation labs. The topic of in-circuit emulation is reserved for a more advanced emulation/analysis class which customers can attend.

STUDENT NOTES:

□ Introduction to Emulation



EMULATION

Emulation	*vs.* *Simulation*
-ambition to equal or surpass another	-an appratus that generates test conditions approximating actual or operational conditions
-real time	-non-real time
-software runs on processor	-software runs on host

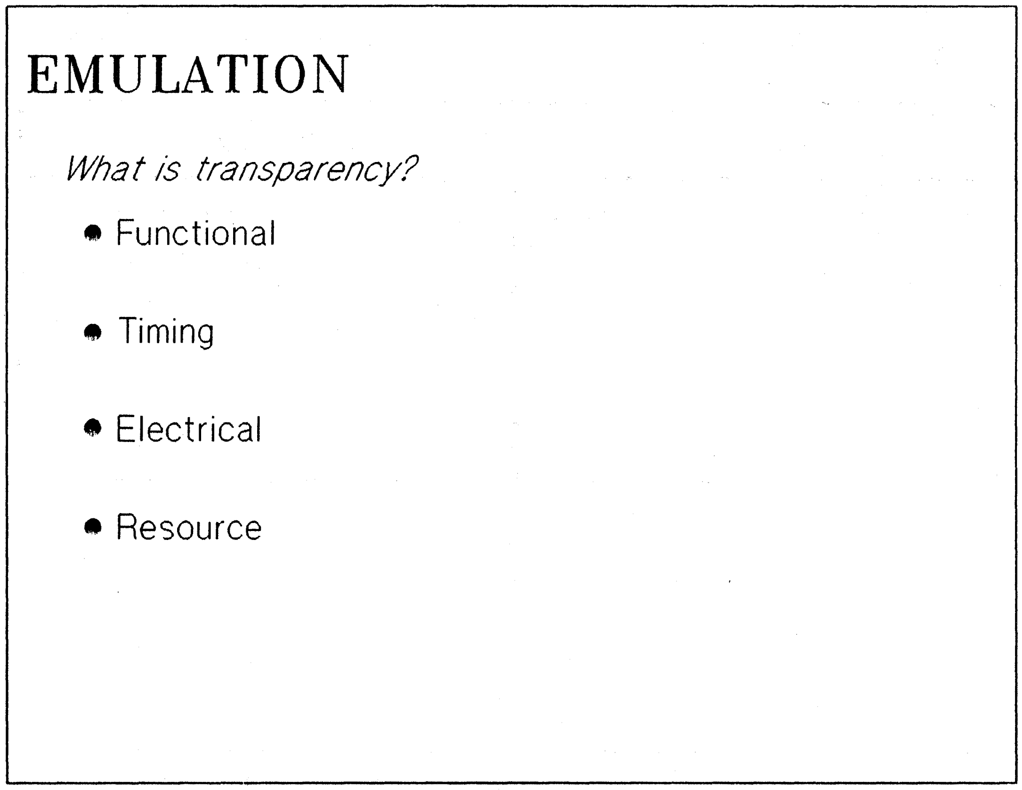
EM1 1.20 © 1987 Hewlett-Packard Company

NOTES:

non-real time execution: this applies to the idea of simulation. All the features of the emulator are available when running in this mode.

real time execution: this applies to the idea of emulation. The emulator will run the user's code without interruptions from the emulation monitor program, thus some of the display / modify commands of the emulation monitor are prohibited.

STUDENT NOTES:



EMULATION

What is transparency?

- Functional
- Timing
- Electrical
- Resource

EM1 1.25 ○ 1987 Hewlett-Packard Company

MANUAL: pages 1-7 thru 1-8 8 and 16 Bit Emulation Reference

NOTES:

`transparency:` the characteristic of the emulator that the function, signal quality, signal timing, loading, drive capacity, and other factors at the plug-in pins should be indistinguishable from the actual processor.

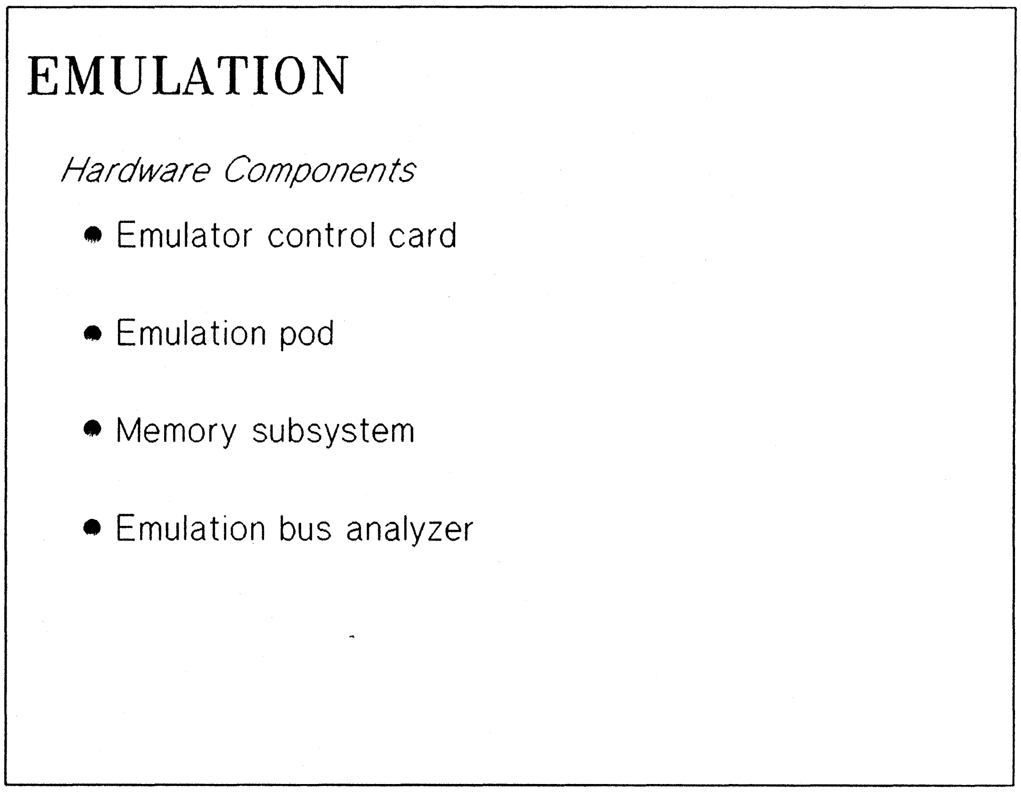
`functional transparency:` the ability of the emulator to function exactly as the processor would.

`timing transparency:` the ability of the emulator to preserve the timing relationships between signals of the actual processor and not to introduce excessive timing signal skew.

`electrical transparency:` the ability of the emulator to preserve the electrical characteristics (loading) of the actual processor.

`resource transparency:` the ability of the emulator to allow the user full access to all the processors resources, i.e. interrupt vectors, I/O ports, counters, timers, etc.

STUDENT NOTES:



EMULATION

Hardware Components

- Emulator control card
- Emulation pod
- Memory subsystem
- Emulation bus analyzer

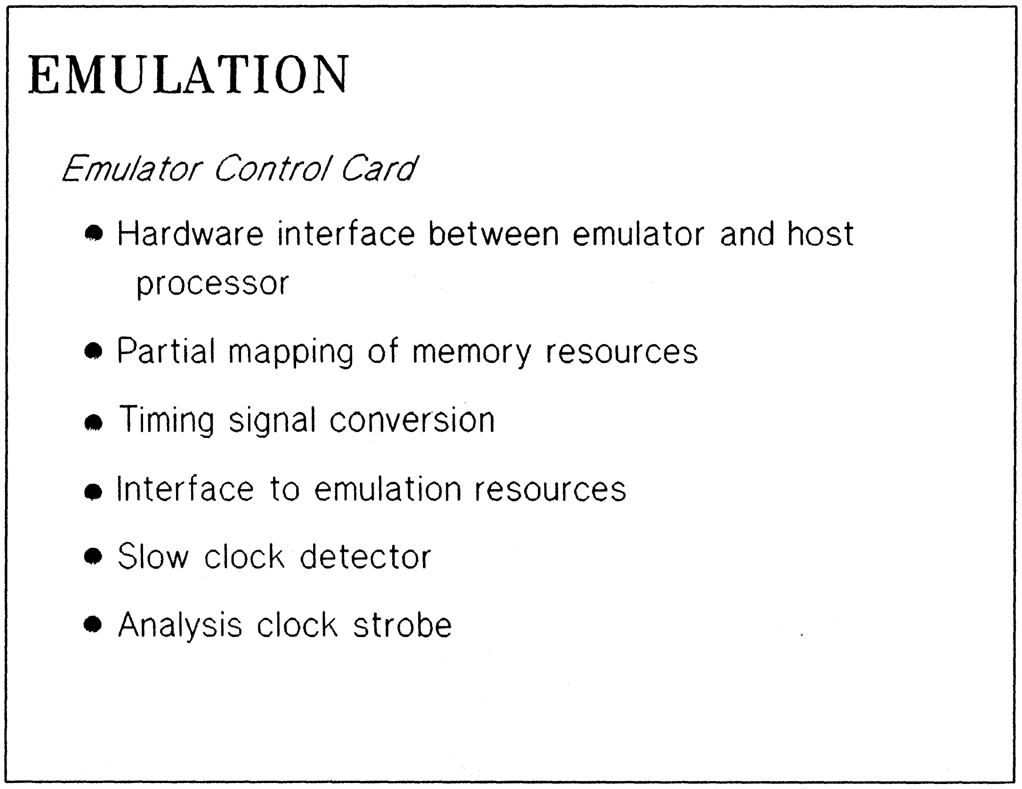
EM1 1.35 ○ 1987 Hewlett-Packard Company

MANUAL: pages 4-1 thru 4-5 8 and 16 Bit Emulation Reference

NOTES:

The memory subsystem of an emulation module may consist of from one to several boards depending upon memory requirements and space limitations in the cardcage.

STUDENT NOTES:



EMULATION

Emulator Control Card

- Hardware interface between emulator and host processor
- Partial mapping of memory resources
- Timing signal conversion
- Interface to emulation resources
- Slow clock detector
- Analysis clock strobe

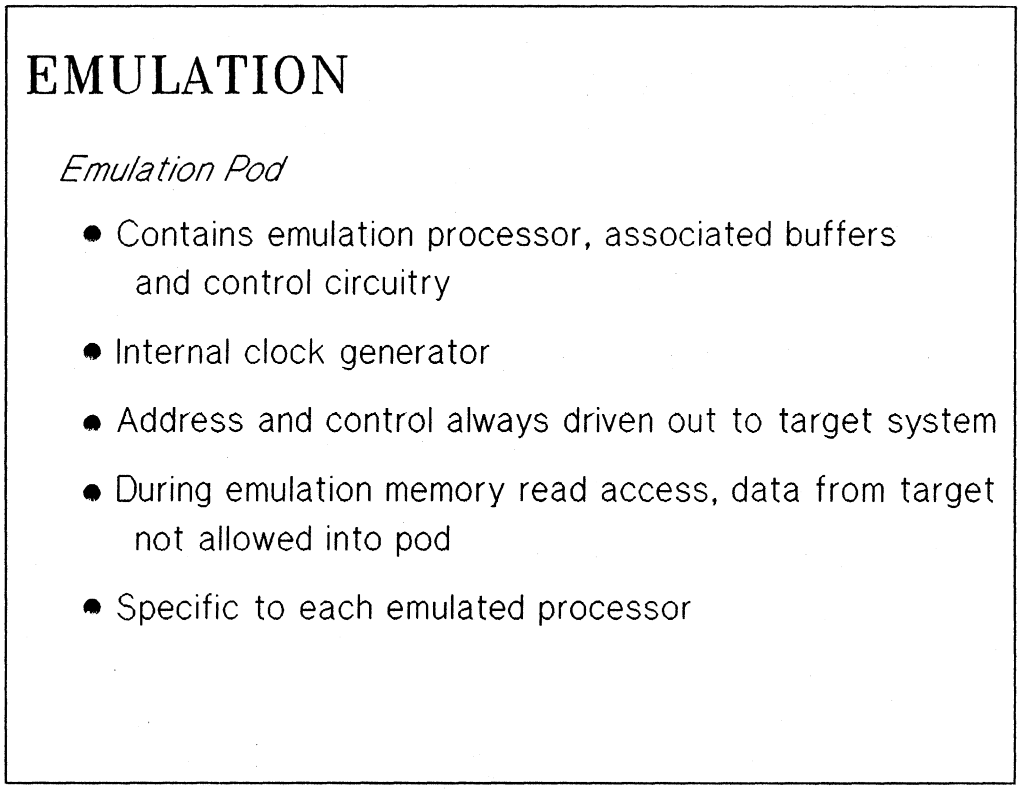
EMI 1.40

NOTES:

The emulator control card controls the interaction between the 64000-UX software and the emulation hardware.

STUDENT NOTES:

Introduction to Emulation



EMULATION

Emulation Pod

- Contains emulation processor, associated buffers and control circuitry
- Internal clock generator
- Address and control always driven out to target system
- During emulation memory read access, data from target not allowed into pod
- Specific to each emulated processor

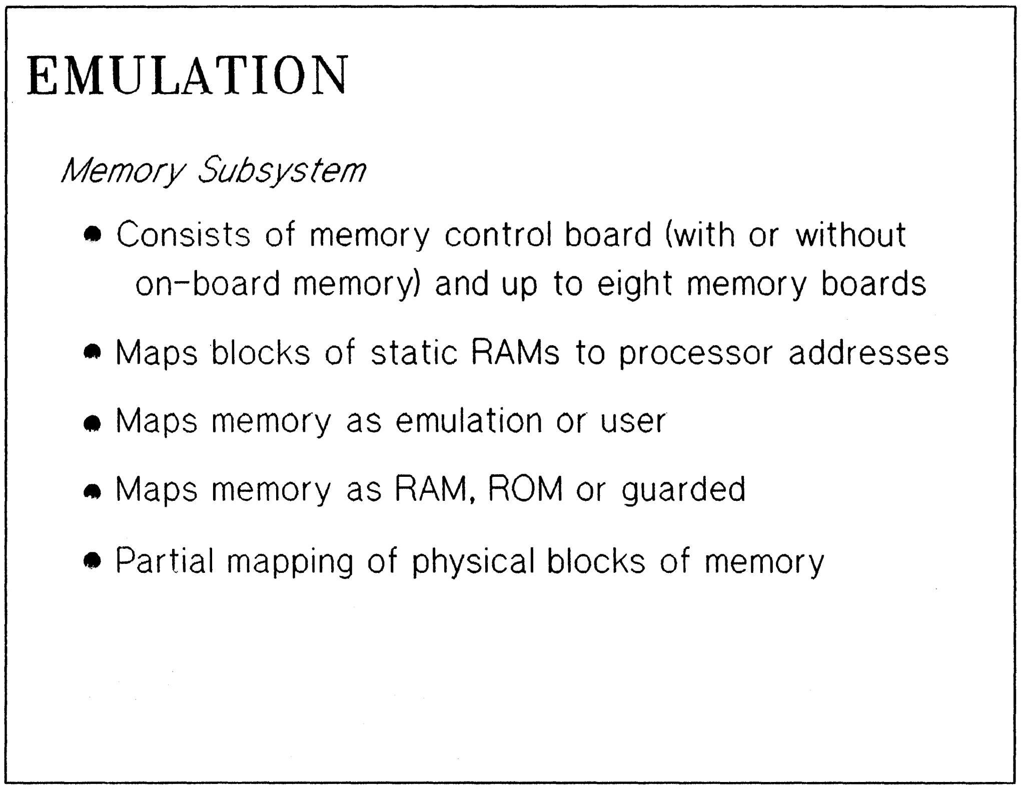
EMI 1.45 © 1987 Hewlett-Packard Company

NOTES:

The emulation pod contains the processor that executes the code and plugs into the target system processor socket.

The exceptions to the address and control information being driven out to the target system are during a processor reset or while the internal clock of the emulator is being utilized.

STUDENT NOTES:



EMULATION

Memory Subsystem

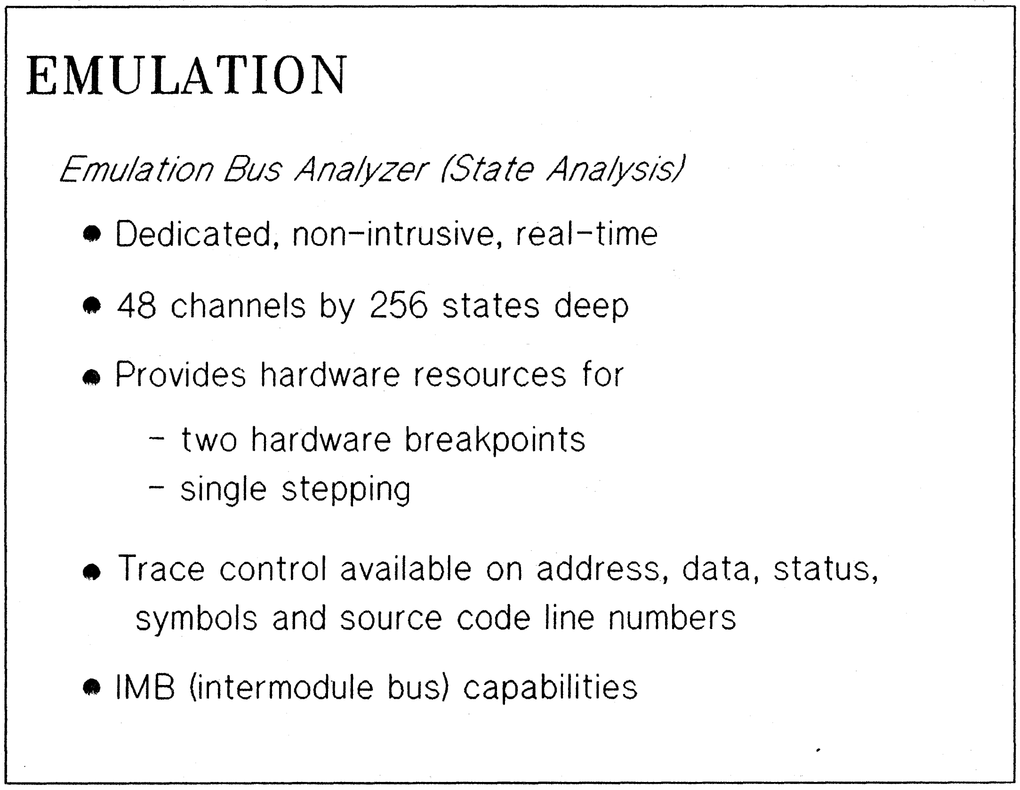
- Consists of memory control board (with or without on-board memory) and up to eight memory boards
- Maps blocks of static RAMs to processor addresses
- Maps memory as emulation or user
- Maps memory as RAM, ROM or guarded
- Partial mapping of physical blocks of memory

EMI 1.50

NOTES:

The memory subsystem consists of memory control board and one to eight emulation memory boards each being 32K, 64K, or 128K bytes per board

STUDENT NOTES:



EMULATION

Emulation Bus Analyzer (State Analysis)

- Dedicated, non-intrusive, real-time
- 48 channels by 256 states deep
- Provides hardware resources for
  - two hardware breakpoints
  - single stepping
- Trace control available on address, data, status, symbols and source code line numbers
- IMB (intermodule bus) capabilities

EM1 1.55

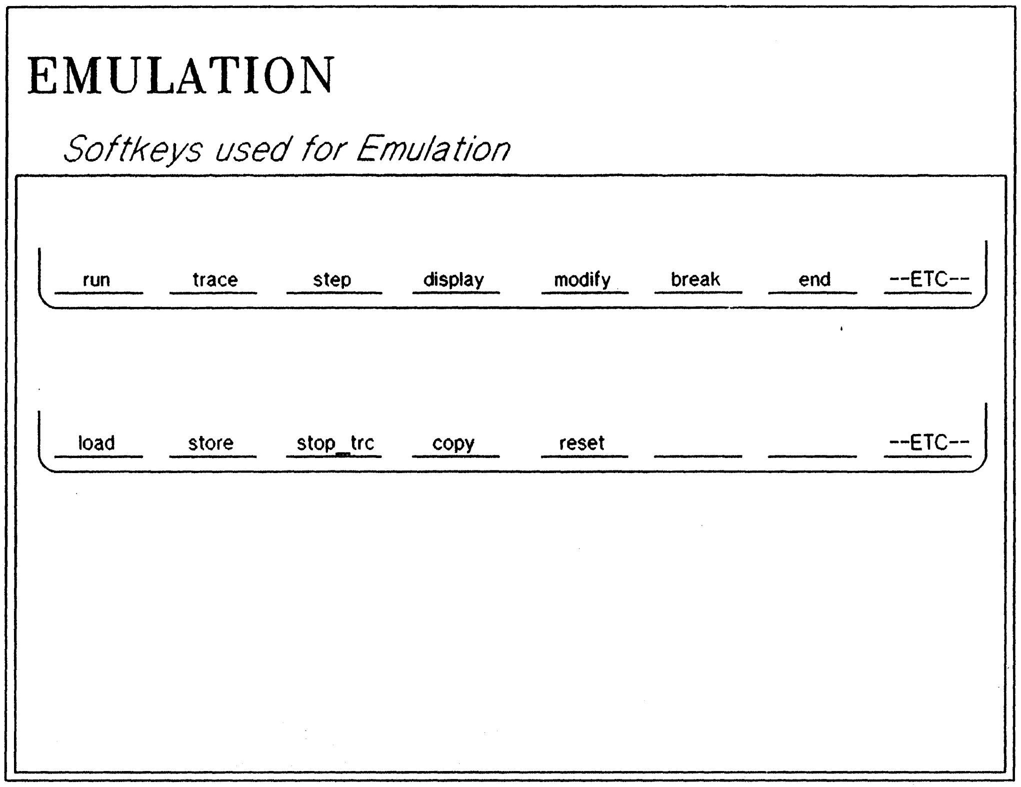
MANUAL: pages 4-6 thru 4-7 8 and 16 Bit Reference Manual

NOTES:

The emulation bus analyzer will give the user the required state analysis and is tightly coupled to the emulator, i.e. functions as the IMB source for the emulator.

STUDENT NOTES:

Introduction to Emulation



EMULATION

Softkeys used for Emulation

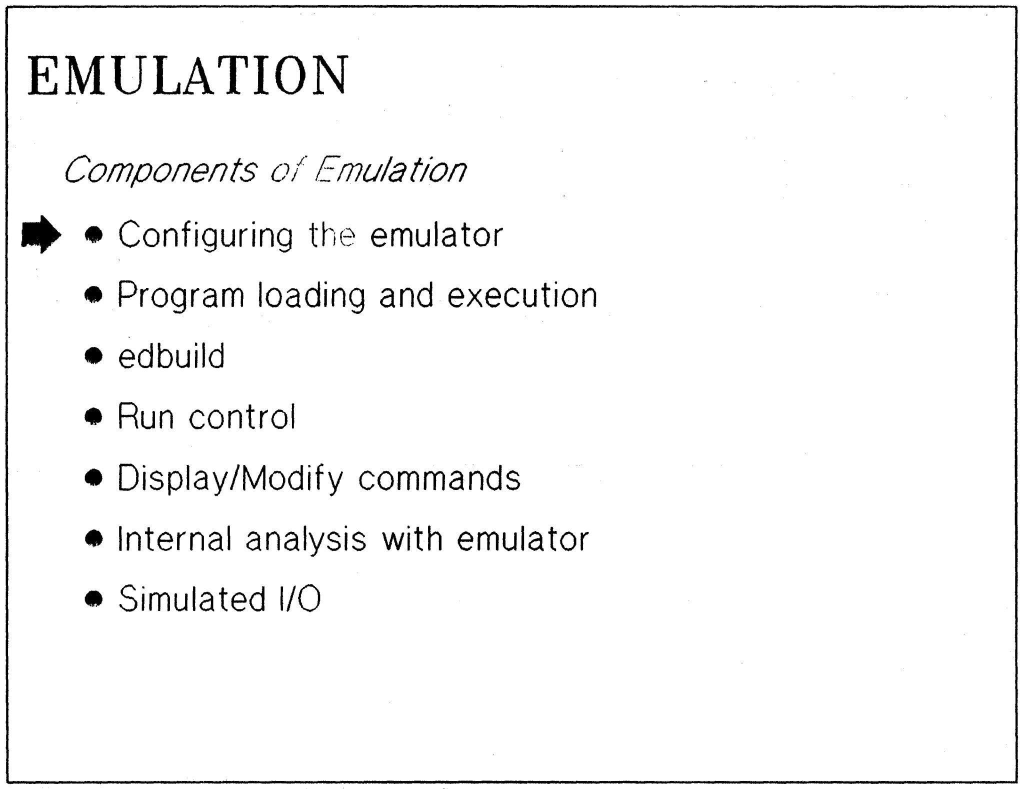
```
run   trace   step   display   modify   break   end   --ETC--

load  store   stop_trc  copy   reset                 --ETC--
```

EM1 1.65 © 1987 Hewlett-Packard Company

NOTES:

A quick look at the available softkeys for a typical emulator. These allow access to the emulation and internal analysis functions of an emulation module.

STUDENT NOTES:



EMULATION

Components of Emulation

- ➡ Configuring the emulator
- Program loading and execution
- edbuild
- Run control
- Display/Modify commands
- Internal analysis with emulator
- Simulated I/O

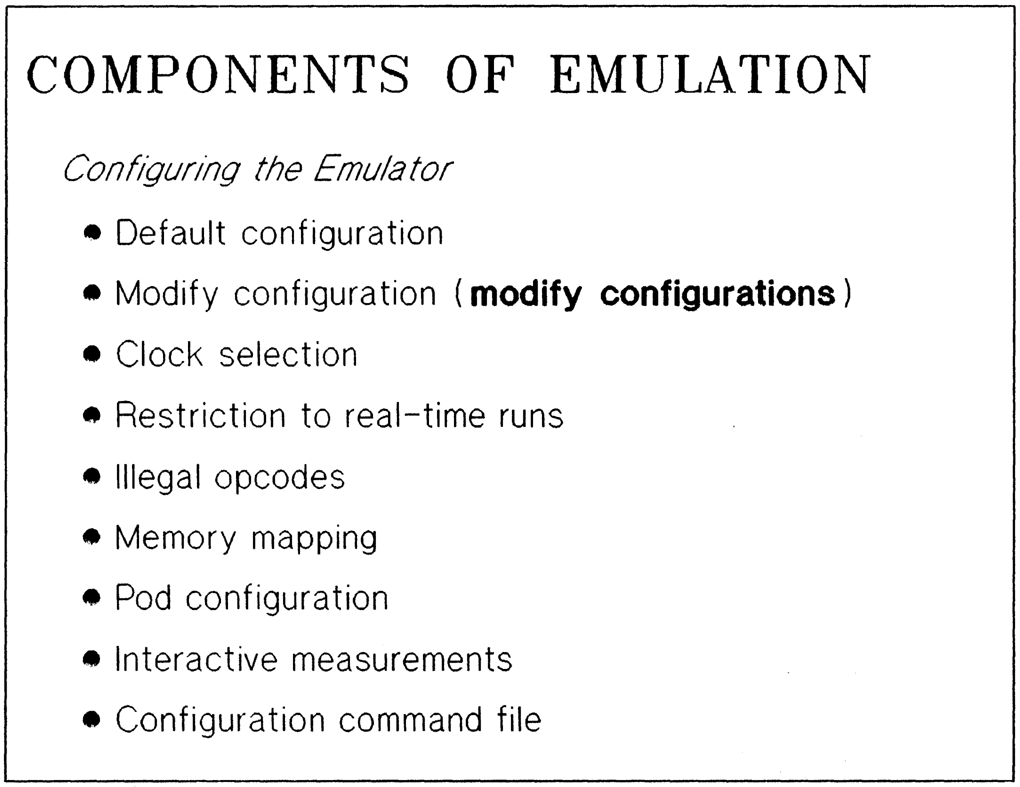
EM1 1.60 © 1987 Hewlett-Packard Company

NOTES:

The components of emulation listed above are part of every 64000-UX emulator.

Let's take a look at each one of these separately...

STUDENT NOTES:



COMPONENTS OF EMULATION

Configuring the Emulator

- Default configuration
- Modify configuration (**modify configurations**)
- Clock selection
- Restriction to real-time runs
- Illegal opcodes
- Memory mapping
- Pod configuration
- Interactive measurements
- Configuration command file

EMI 1.70 ○ 1987 Hewlett-Packard Company

MANUAL: pages 2-1 thru 2-8 8 and 16 Bit Emulation Reference

NOTES:

A 64000-UX emulator is supplied with a default emulation configuration which is loaded and used unless the user specifies another configuration.

The user may also elect to "modify" the existing configuration and save the new configuration to be used at a later date.

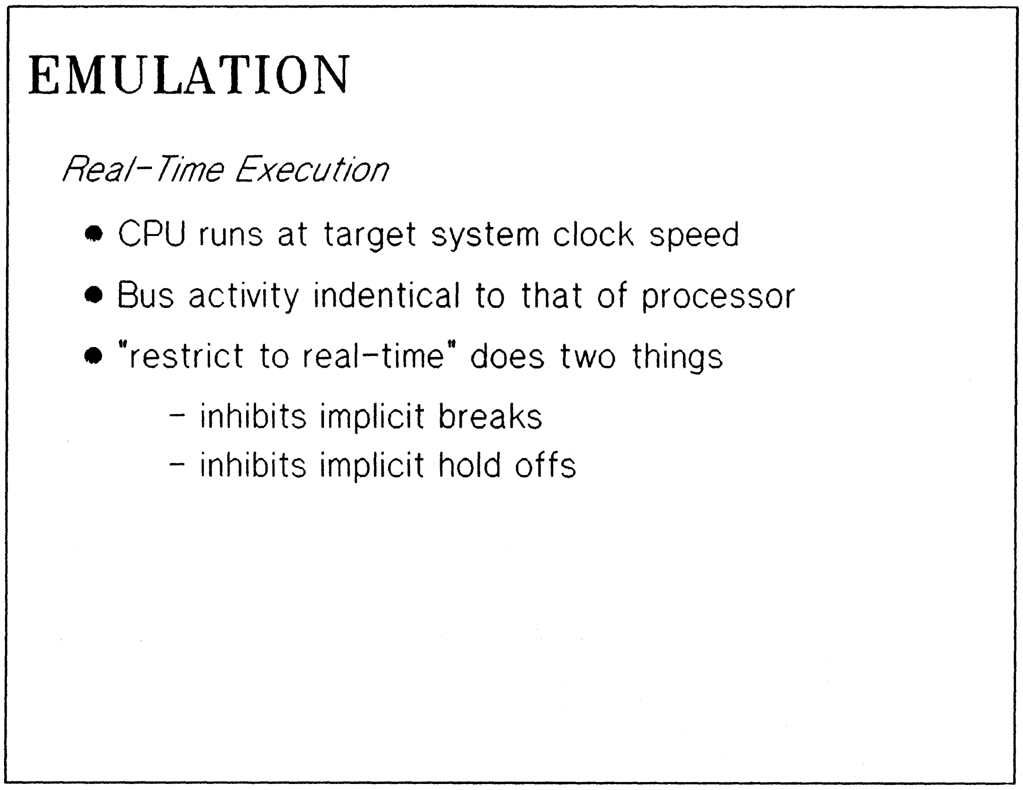
The internal/external clock selection determines if the emulation processor's clock source will come from the emulation pod or the user's target system. The emulator gives the user a choice of breaking into the monitor if an illegal opcode is detected on the processor data bus.

The topic of emulation memory mapping will be covered in detail later in this section.

After answering all the configuration questions, the user has the chance to name and save the emulation configuration just setup. If created, the new emulation configuration file can be loaded every time the emulator is used, thus saving a great amount of time.

STUDENT NOTES:

☐ Introduction to Emulation



EMULATION

Real-Time Execution

- CPU runs at target system clock speed
- Bus activity indentical to that of processor
- "restrict to real-time" does two things
  - inhibits implicit breaks
  - inhibits implicit hold offs

EM1 1.75 ○ 1987 Hewlett-Packard Company

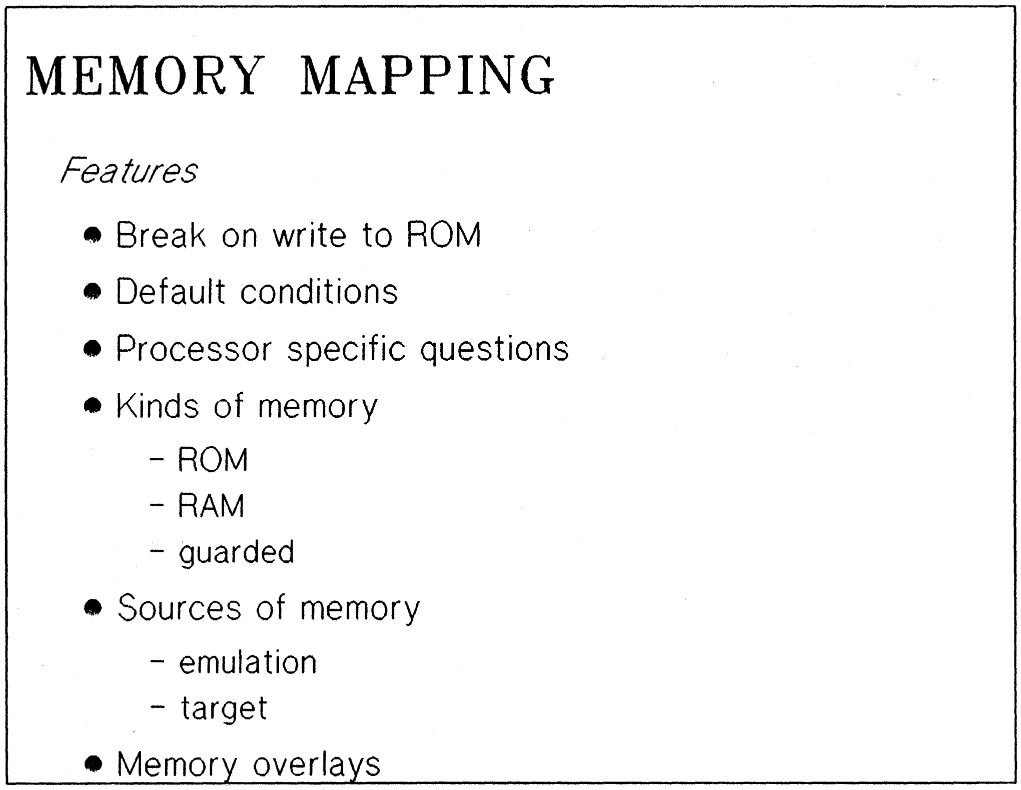
MANUAL: page 2-3 8 and 16 Bit Emulation Reference

MANUAL: pages 2-2 thru 2-4 8 and 16 Bit Emulation Reference

NOTES:

real-time execution: is an advanced emulation topic and is better covered in an advanced emulation class.

STUDENT NOTES:



MEMORY MAPPING

Features

- Break on write to ROM
- Default conditions
- Processor specific questions
- Kinds of memory
  - ROM
  - RAM
  - guarded
- Sources of memory
  - emulation
  - target
- Memory overlays

EMI 1.80 ◯ 1987 Hewlett-Packard Company

MANUAL: pages 2-4 thru 2-8 8 and 16 Bit Emulation Reference

NOTES:

The emulator can be configured to break on the detection of a memory write to an address location reserved as ROM space.

Some emulators may have processor specific questions such as the one which appears for the 68000 concerning the number of significant address bits.

The address space of the emulator can now be mapped as one of the following types of memory:
- ROM
- RAM
- guarded

These types of memory can also be defined to reside in one of two places:
- emulation
- target system

NOTE: as an example the 68000 foreground monitor software must always be defined as "emulation RAM". The appropriate amount of address space must also be reserved for this purpose, i.e. the 68000 monitor program requires a 4 KBytes block of memory.

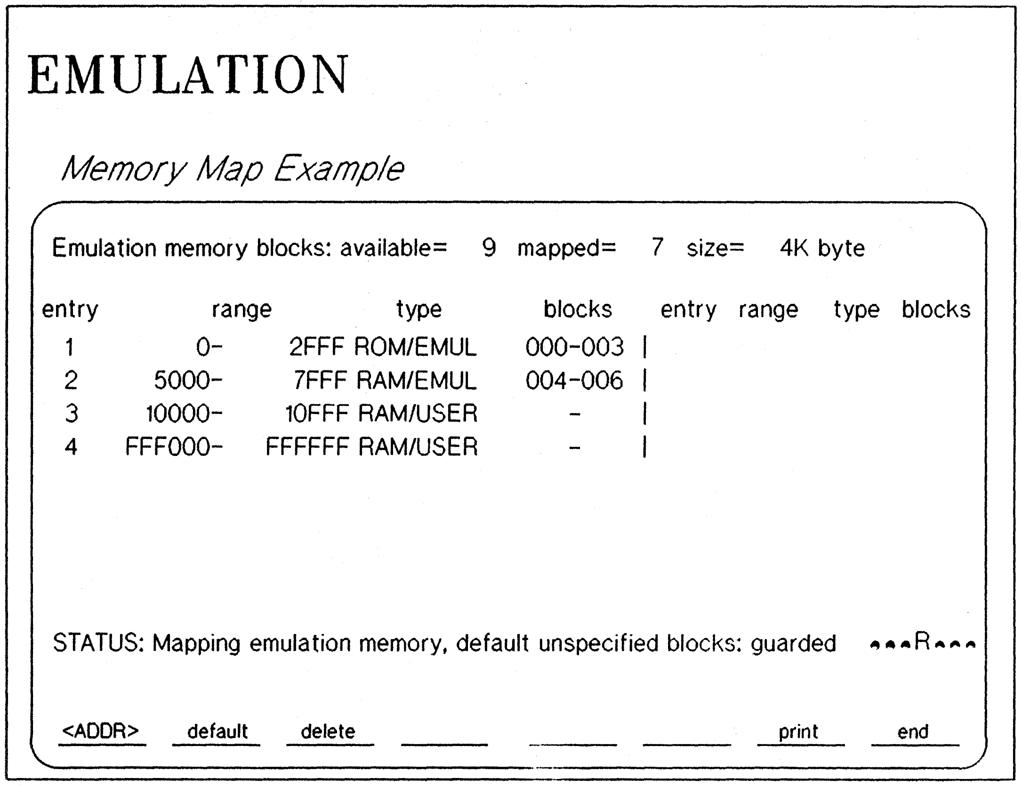
□ Introduction to Emulation

Any memory space not mapped will be mapped to the memory default. For instance, any memory not defined would be mapped as "guarded" if the memory default was defined as "guarded".

Memory overlays allow the user to define one block of memory to respond to two or more sets of addresses. This can be useful to help reduce the amount of required target system memory during development if identical code were to reside in two separate memory areas.

STUDENT NOTES:

☐ Introduction to Emulation



EMULATION

Memory Map Example

```
Emulation memory blocks: available=   9  mapped=   7  size=   4K byte

entry         range            type        blocks   entry  range   type  blocks
  1          0-       2FFF ROM/EMUL      000-003 |
  2       5000-       7FFF RAM/EMUL      004-006 |
  3      10000-      10FFF RAM/USER         -    |
  4    FFF000-     FFFFFF RAM/USER         -    |


STATUS: Mapping emulation memory, default unspecified blocks: guarded    ...R...

<ADDR>   default   delete   ______   ______   ______   print   end
```

EM1 1.85 © 1987 Hewlett-Packard Company

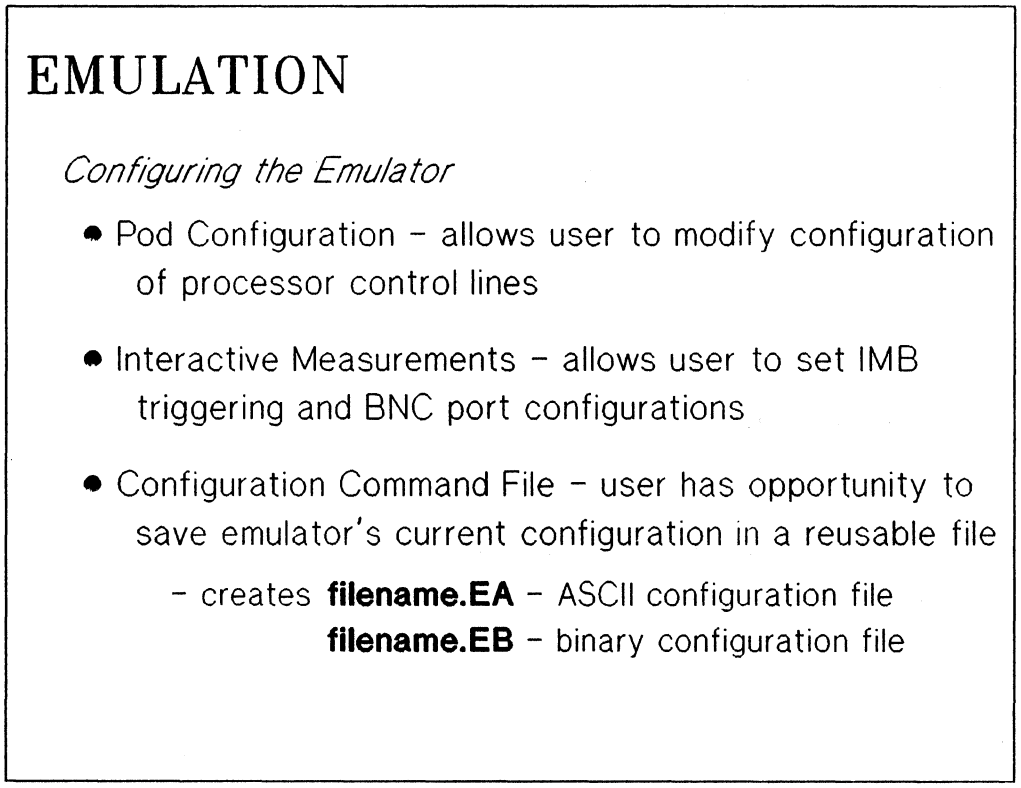
NOTES:

The general screen format of the memory mapper is shown in the above slide.

The screen will list the range of the address mapping as well as the type of memory the space is mapped to. It also keeps track of the number of memory blocks which have been allocated.

STUDENT NOTES:

Introduction to Emulation



EMULATION

Configuring the Emulator

- Pod Configuration – allows user to modify configuration of processor control lines
- Interactive Measurements – allows user to set IMB triggering and BNC port configurations
- Configuration Command File – user has opportunity to save emulator's current configuration in a reusable file
  - creates **filename.EA** – ASCII configuration file
    filename.EB – binary configuration file

EM1 1.90 © 1987 Hewlett-Packard Company

MANUAL: Chapter 5 Emulation Configuration 68000 Emulation Manual

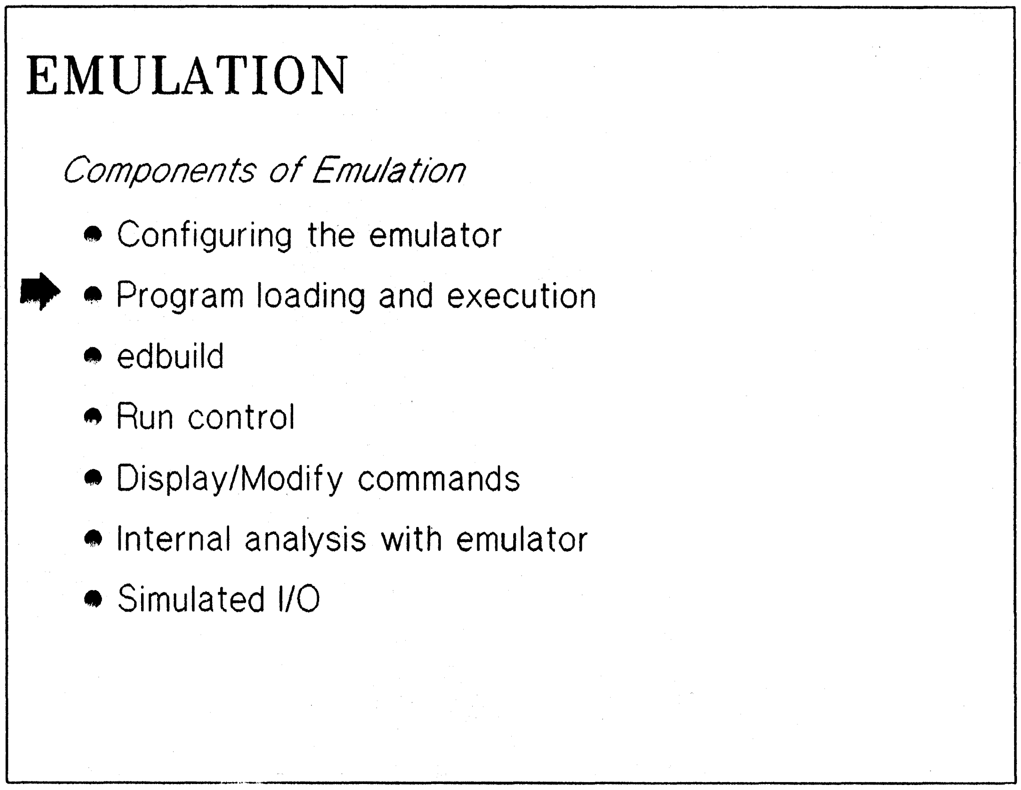
NOTES:

pod configuration: these questions are specific to every emulator and will not be covered for the 68000 due to the complex nature of them.

interactive measurements: this portion of the configuration is to allow the user to define how his IMB is going to be used between other analyzers and whether or not any signals will be accepted and configured for the BNC ports on the cardcage.

configuration command file: saves the emulators current configuration in a binary file which is loaded into the emulator to set the configuration and also saves the configuration in an ASCII file which is editable by the user.

STUDENT NOTES:



EMULATION

Components of Emulation

- Configuring the emulator
- Program loading and execution
- edbuild
- Run control
- Display/Modify commands
- Internal analysis with emulator
- Simulated I/O

EMI 1.93

STUDENT NOTES:

COMPONENTS OF EMULATION

Program Loading

- Loading the absolute
  - emulation memory
  - target memory
- Foreground monitor must be loaded before loading user memory
  - this function is performed automatically if the monitor is linked with the user's relocatables

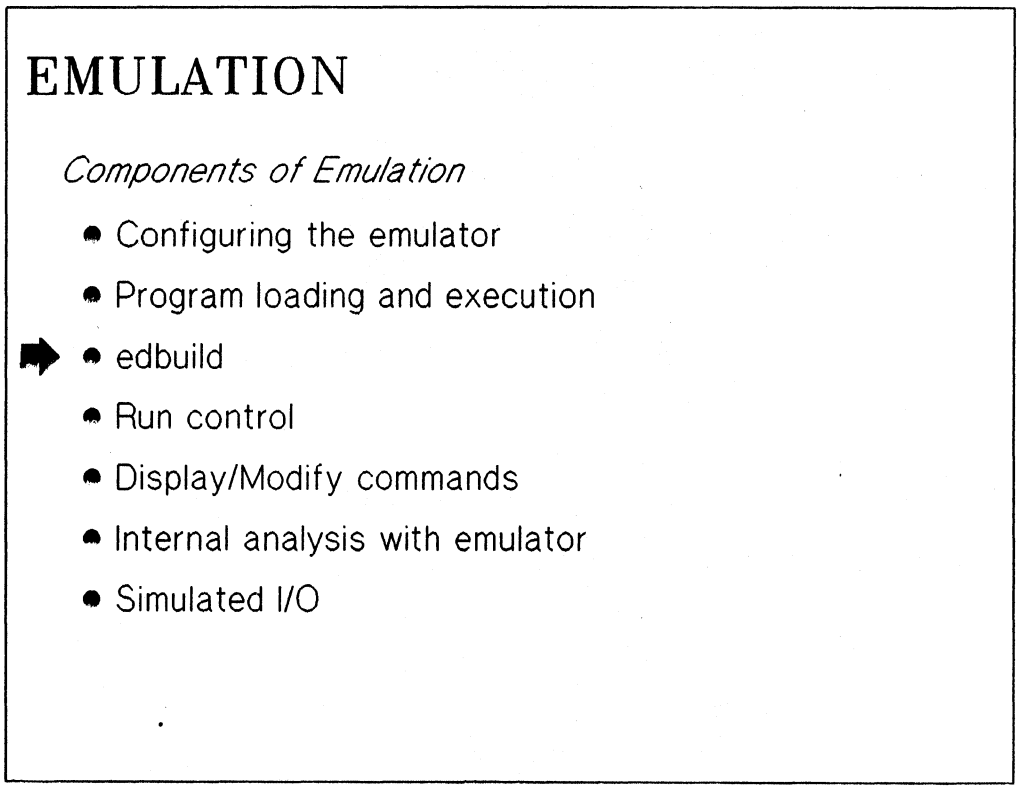
EM1 1.95 © 1987 Hewlett-Packard Company

NOTES:

When a "load memory" command is issued, the memory map in the emulation configuration determines where the absolute code is to be loaded; either in emulation memory or target system memory. The emulation control board contains the emulation memory mapper which takes care of this feature.

STUDENT NOTES:

□ Introduction to Emulation



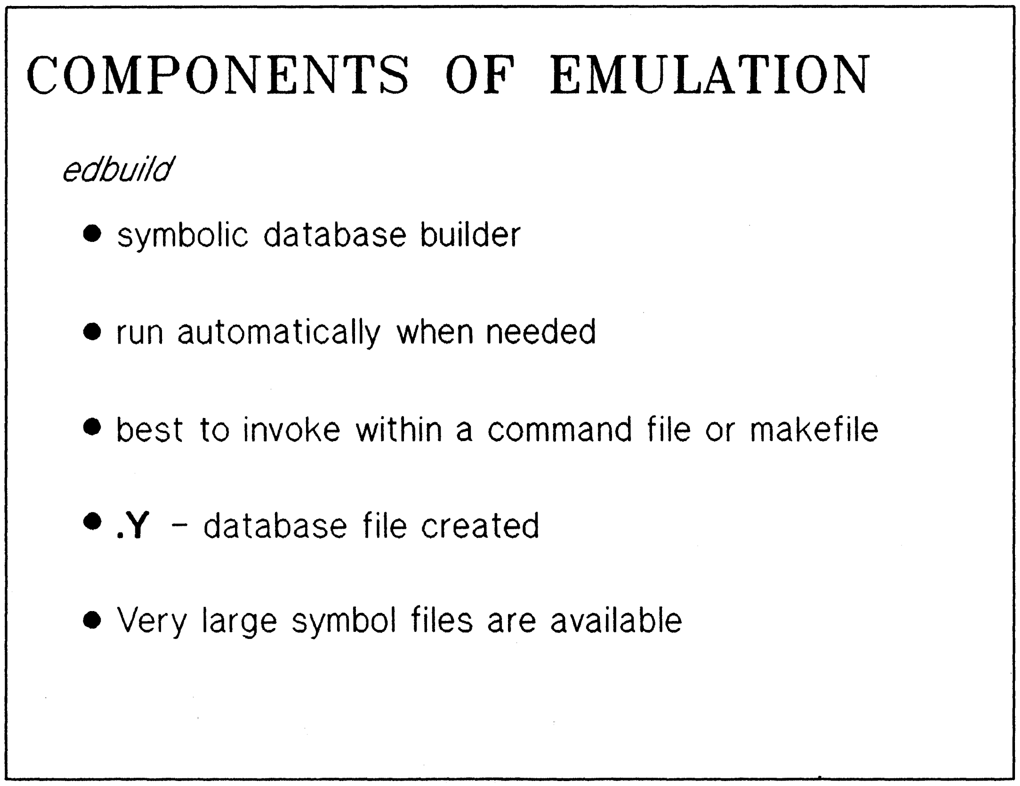
EMULATION

Components of Emulation

- Configuring the emulator
- Program loading and execution
- ➡ edbuild
- Run control
- Display/Modify commands
- Internal analysis with emulator
- Simulated I/O

EM1 1.98

STUDENT NOTES:



COMPONENTS OF EMULATION

edbuild

- symbolic database builder
- run automatically when needed
- best to invoke within a command file or makefile
- **.Y** - database file created
- Very large symbol files are available

EMI 2.00

NOTES:

edbuild: reads files produced by the language system and produces a database used by 64000-UX products that provide symbolic access, such as emulation and analysis.

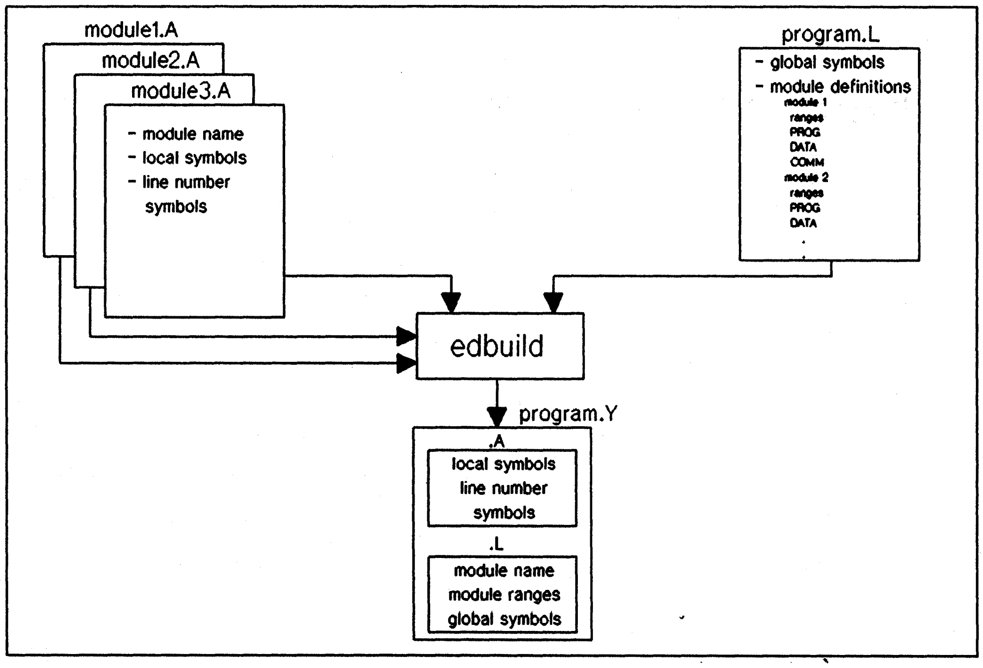
Edbuild is run automatically whenever an access from an emulator or analyzer requires symbolic information. The builder creates the database file only as changes are made to the source code.

Although edbuild will run automatically, it is most efficient to invoke in a makefile so that it is only run during compilation and linking and only as needed.

STUDENT NOTES:

□ Introduction to Emulation

EDBUILD



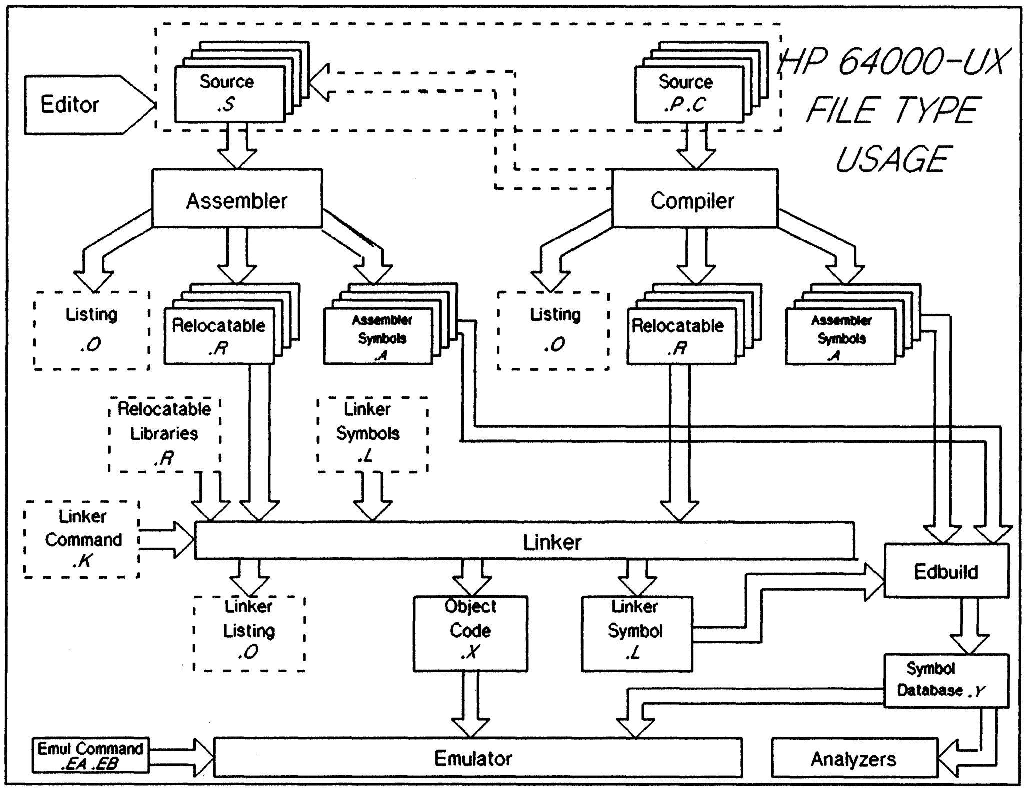
EM1 2.05

NOTES:

The above diagram shows the input files to the edbuild program and the type of information contained in the ".Y" file.

STUDENT NOTES:

□ Introduction to Emulation



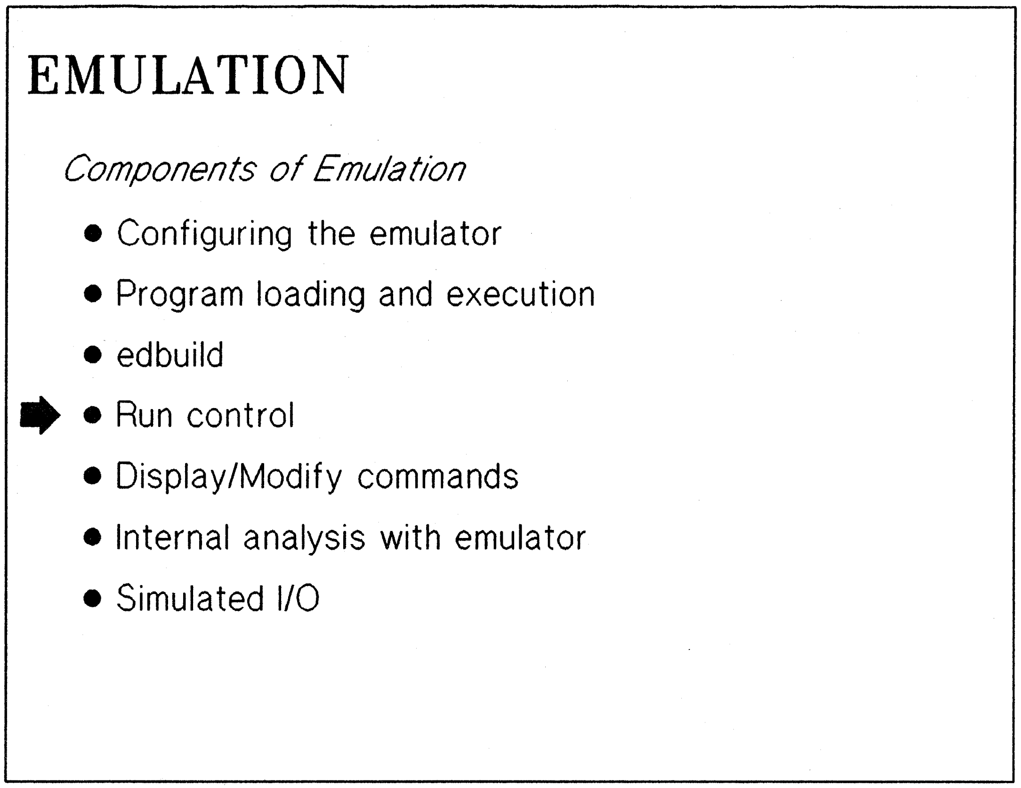
EM1 2.10 © 1987 Hewlett-Packard Company

NOTES:

This diagram shows the entire software development process, from creating the source files with an editor to loading the object code into the emulator.

NOTE: all types of files with their representative file extensions are shown.

STUDENT NOTES:



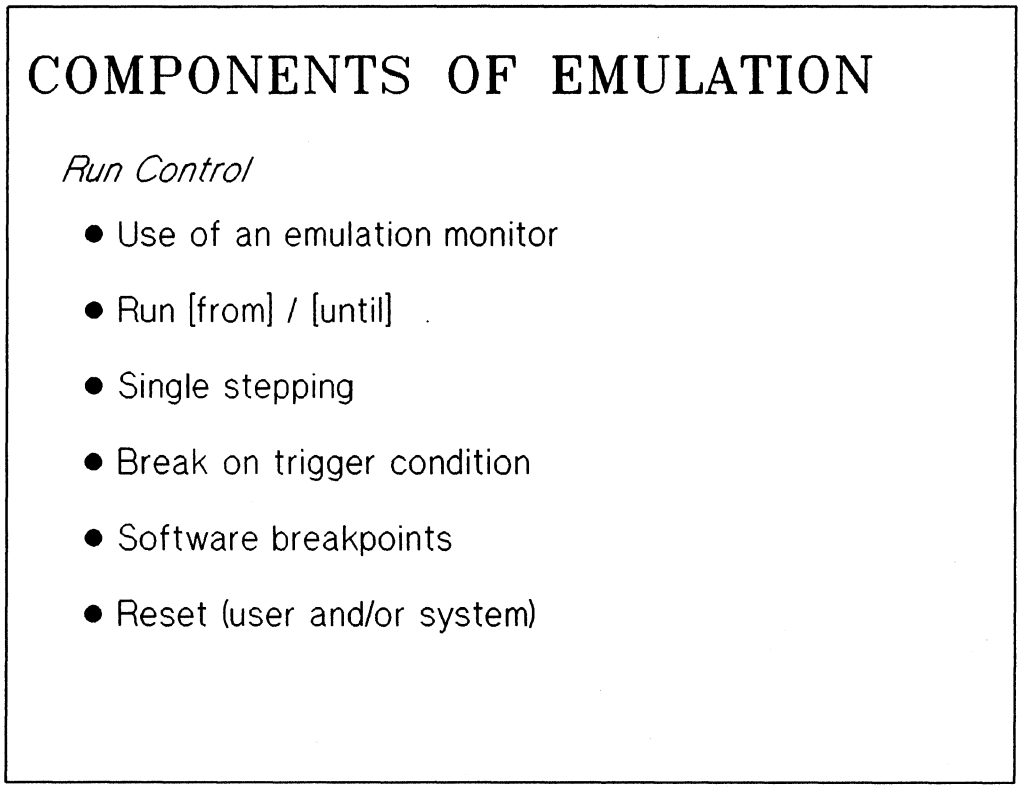
EMULATION

Components of Emulation

- Configuring the emulator
- Program loading and execution
- edbuild
- Run control
- Display/Modify commands
- Internal analysis with emulator
- Simulated I/O

EM1 2.13

STUDENT NOTES:



COMPONENTS OF EMULATION

Run Control

- Use of an emulation monitor
- Run [from] / [until]
- Single stepping
- Break on trigger condition
- Software breakpoints
- Reset (user and/or system)

EM1 2.15

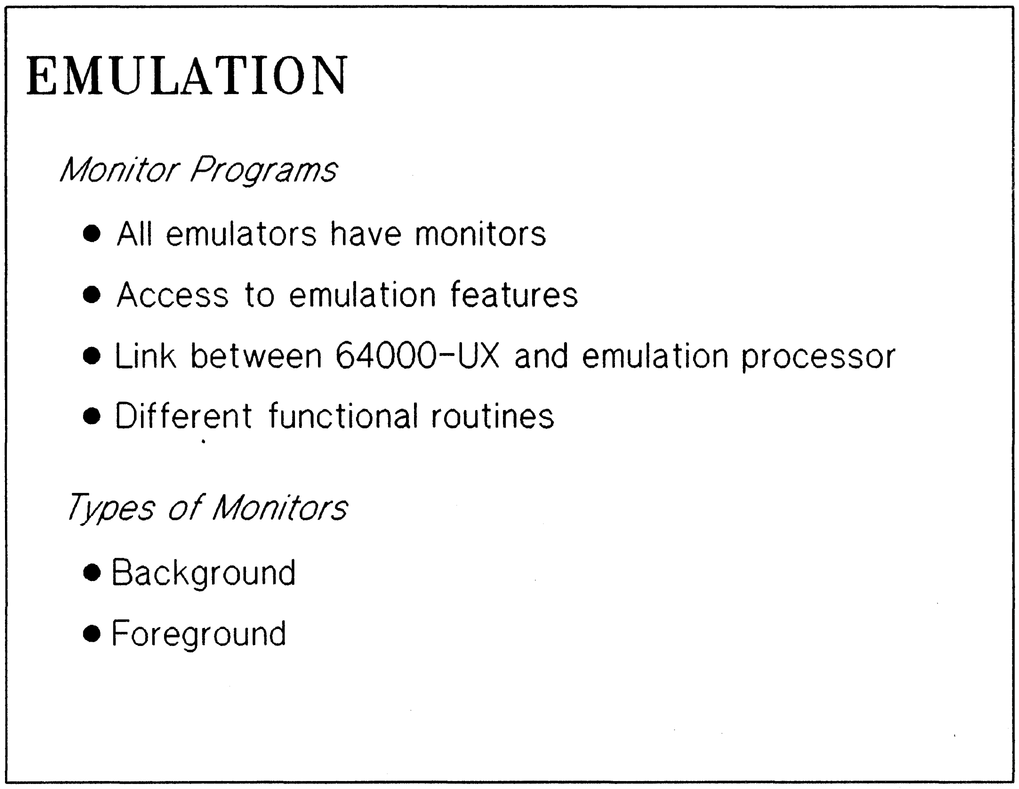
NOTES:

run [from] / [until]: allows the user to run from the transfer address of the absolute code or from any one address to another address inside the address space of the absolute code. *step [number]:* allows the user to single step the processor for one or more instructions. This is useful while displaying the processor's registers to help debug the target system hardware and software.

break: this is used for control transitioning between the user's code and the emulation monitor program.

reset: a processor reset can be issued from the target system or from the softkey command within the foreground monitor of the 68000. This condition will cause the emulation processor and the emulation hardware to be reset.

STUDENT NOTES:



EMULATION

Monitor Programs

- All emulators have monitors
- Access to emulation features
- Link between 64000-UX and emulation processor
- Different functional routines

Types of Monitors

- Background
- Foreground

EM1 2.20

MANUAL: pages 6-1 thru 6-13 68000 Emulation W/Int. Analysis

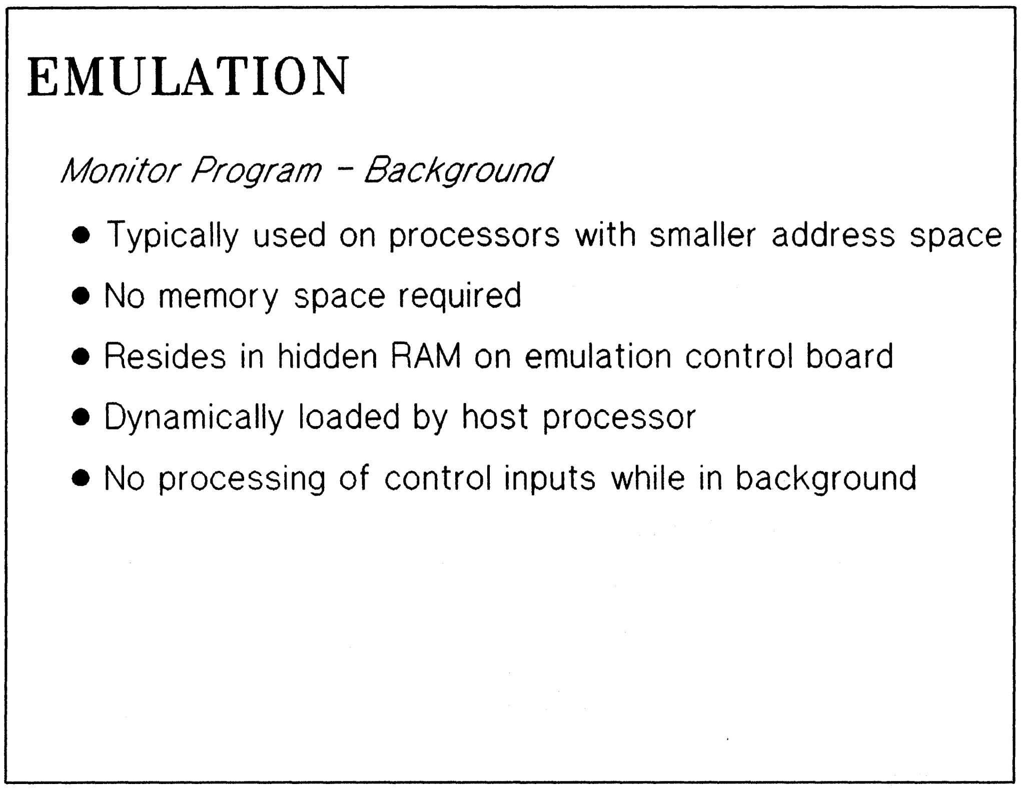
NOTES:

monitor program: the program which seizes control of the user's program and makes available the feature set of the emulator.

background monitor: a 64000-UX system program which is not modifiable by the user. This program seizes control from the user code when a break or reset condition is issued. A background monitor is used by the 6801 emulator.

foreground monitor: a monitor written in the processors assembly code and made available to the user. This type of monitor must be assembled and linked with the user's source code for the emulator to function properly. It gives the emulator the ability to have software breakpoints as well as the ability to access target memory without having to reset the processor. The 68000 emulator utilizes this type of monitor which adds to its functionality.

STUDENT NOTES:



EMULATION

Monitor Program - Background

- Typically used on processors with smaller address space
- No memory space required
- Resides in hidden RAM on emulation control board
- Dynamically loaded by host processor
- No processing of control inputs while in background

EMI 2.25 © 1987 Hewlett-Packard Company

MANUAL: page 4-2 8 and 16 Emulation Reference

NOTES:

When a "background" emulator is "running in the monitor", it is actually running in the memory (RAM) on the emulation control board.

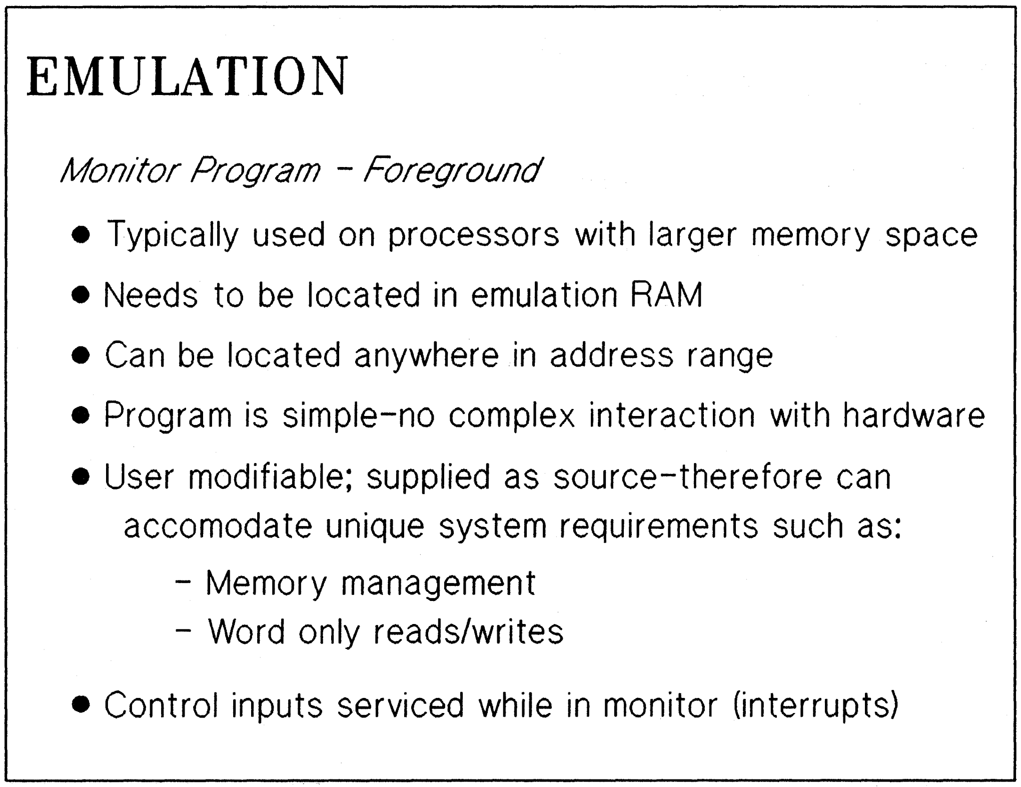
When running in background, the execution of the user's processor is suspended and the processor appears halted to the user system.

The memory space occupied by the monitor is not in the user's address space. This memory is accessible to both the emulation processor and the HP 64000-UX host processor.

Background emulators are usually simple processors with small address spaces and no opcode pre-fetch ability.

STUDENT NOTES:

Introduction to Emulation



EMULATION

Monitor Program – Foreground

- Typically used on processors with larger memory space
- Needs to be located in emulation RAM
- Can be located anywhere in address range
- Program is simple-no complex interaction with hardware
- User modifiable; supplied as source-therefore can accomodate unique system requirements such as:
  - Memory management
  - Word only reads/writes
- Control inputs serviced while in monitor (interrupts)

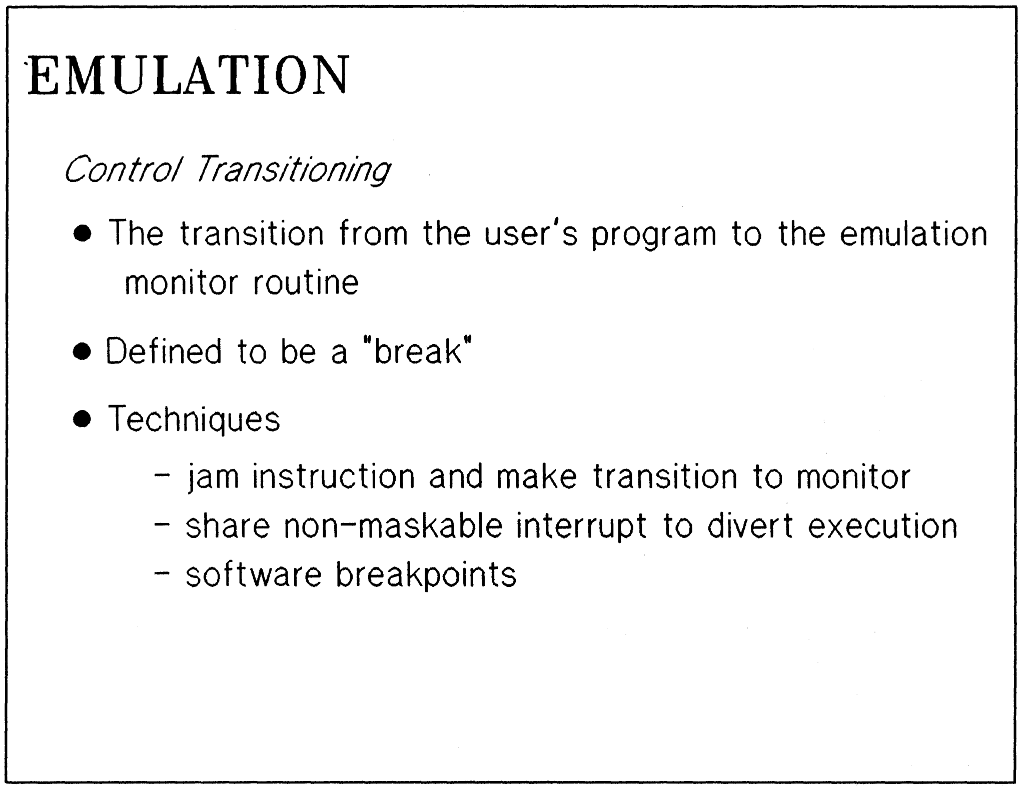
EM1 2.30

NOTES:

A foreground monitor is located in an unused portion of the user's address space. The memory space where the monitor resides must be mapped as emulation memory so that the 64000-UX software and the emulator processor can access it.

The foreground monitor is supplied to the user in source code form. This allows the user to customize the monitor to the specific needs of the project.

STUDENT NOTES:



EMULATION

Control Transitioning

- The transition from the user's program to the emulation monitor routine
- Defined to be a "break"
- Techniques
    - jam instruction and make transition to monitor
    - share non-maskable interrupt to divert execution
    - software breakpoints

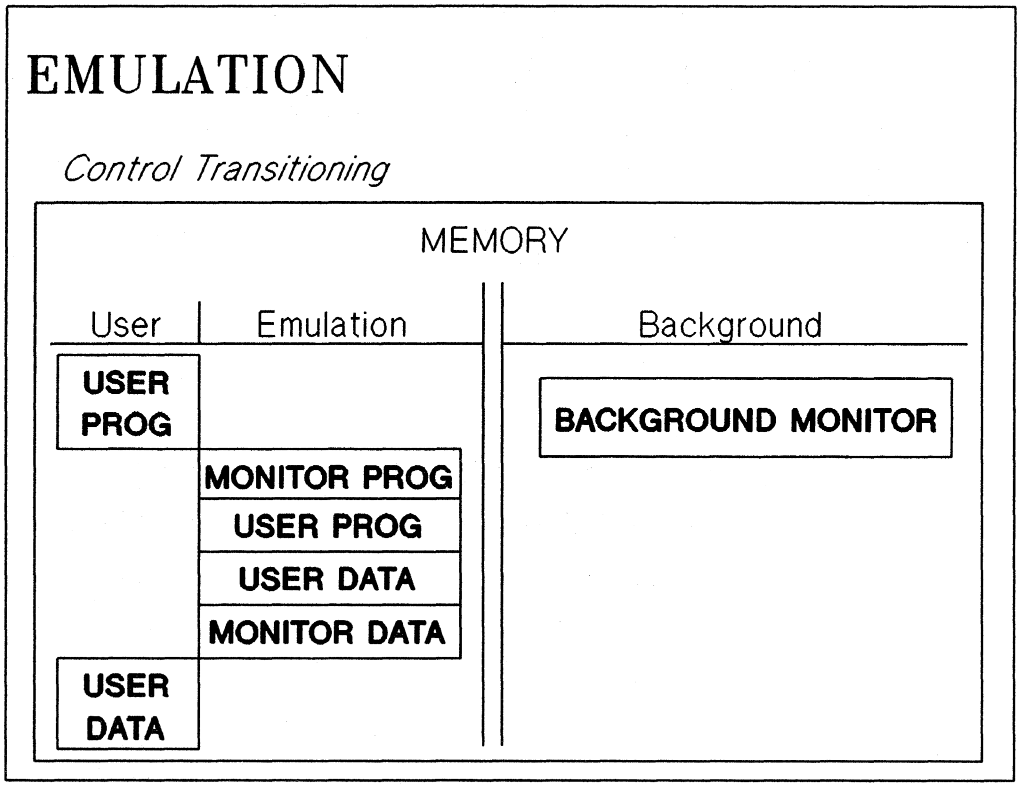
EM1 2.35

NOTES:

break: **the break allows the control of the emulator to be passed from the user's code to the emulation monitor. It is usually implemented in different ways with each processor, due to the processor's resources and control lines.**

STUDENT NOTES:

☐ Introduction to Emulation

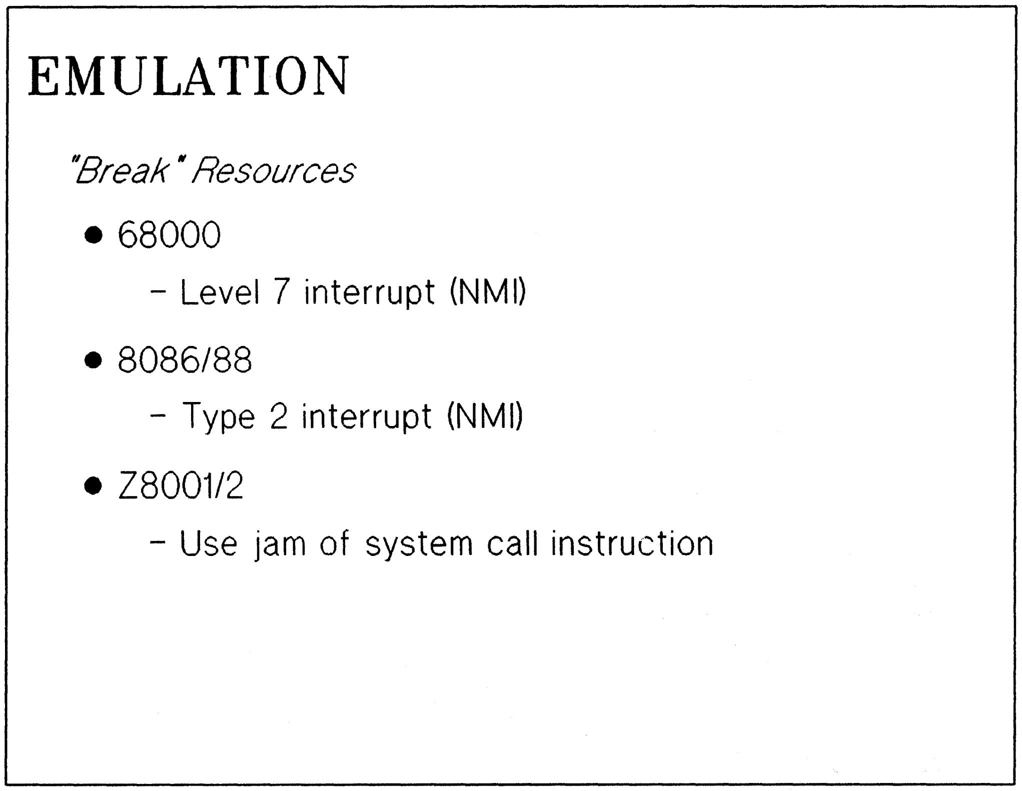


EMI 2.40

NOTES:

This diagram gives a good indication as to where the monitor resides and how the control is "transitioned" between the user code and the monitor.

STUDENT NOTES:



EMULATION

"Break" Resources

- 68000
  - Level 7 interrupt (NMI)
- 8086/88
  - Type 2 interrupt (NMI)
- Z8001/2
  - Use jam of system call instruction

EMI 2.45

NOTES:

Once again, "breaks" to the monitor are implemented differently for different processors depending upon the resources of the processor, such as control lines.

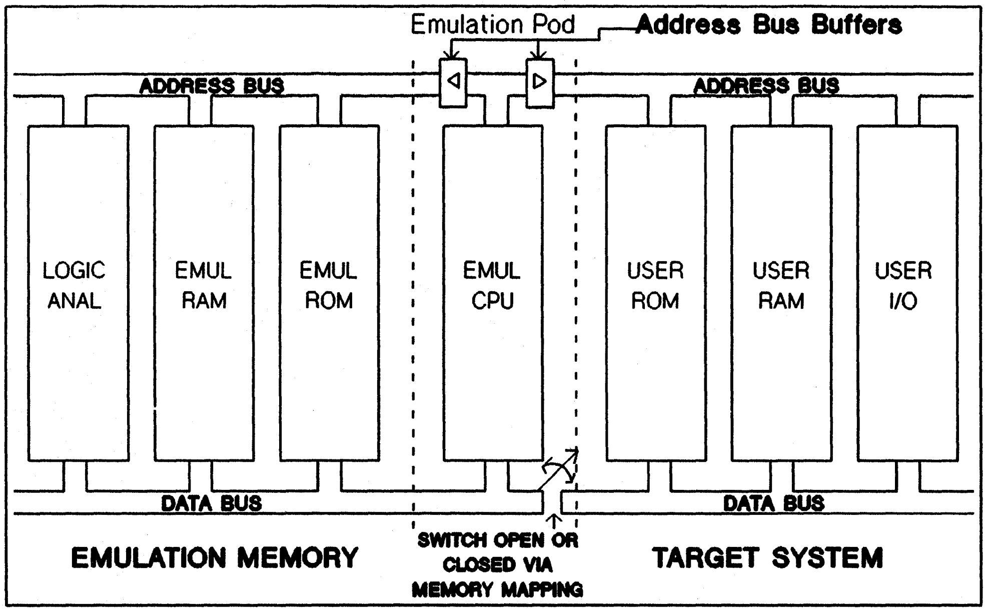
For example: the 68000 uses the Level 7 interrupt (NMI) to transition into the monitor program. This is called resource sharing.

STUDENT NOTES:

☐ Introduction to Emulation

EMULATION

Emulation Subsystem Connection to Target System



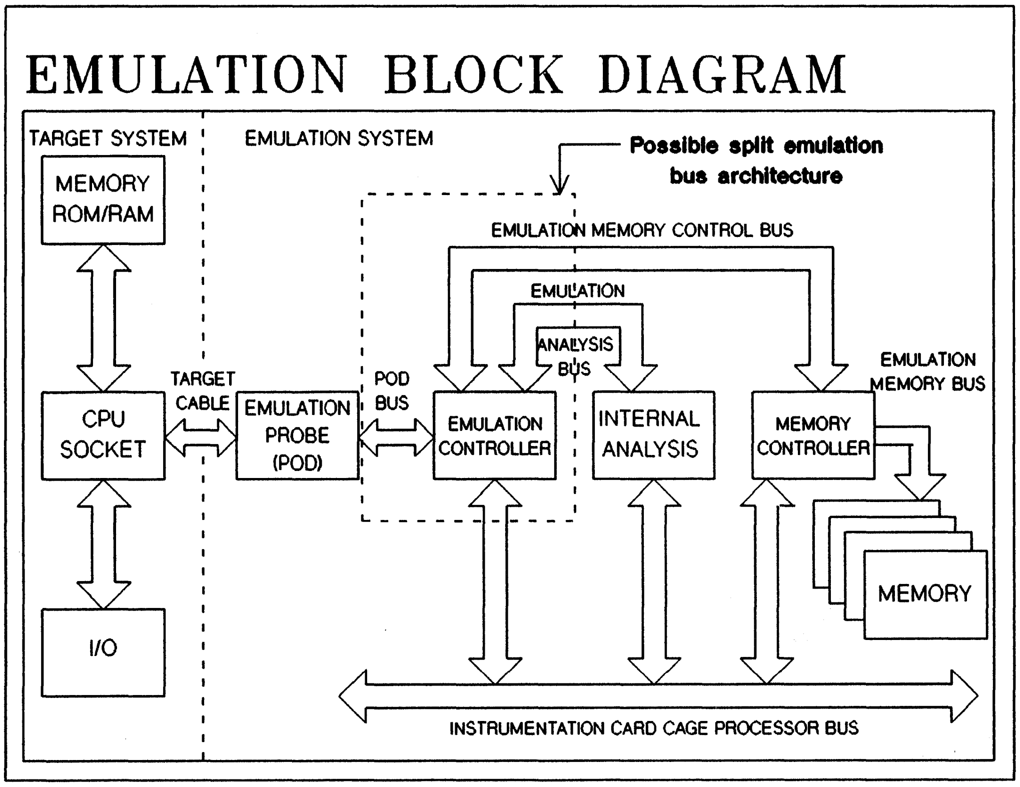
EM1 2.50 © 1987 Hewlett-Packard Company

NOTES:

This diagram gives a good view of the type of connection the emulator has with the user's target system. Pay close attention to the address and data bus structure of the diagram and how these busses can be isolated from the target system if necessary.

STUDENT NOTES:

□ Introduction to Emulation



EMI 2.55 © 1987 Hewlett-Packard Company

NOTES:

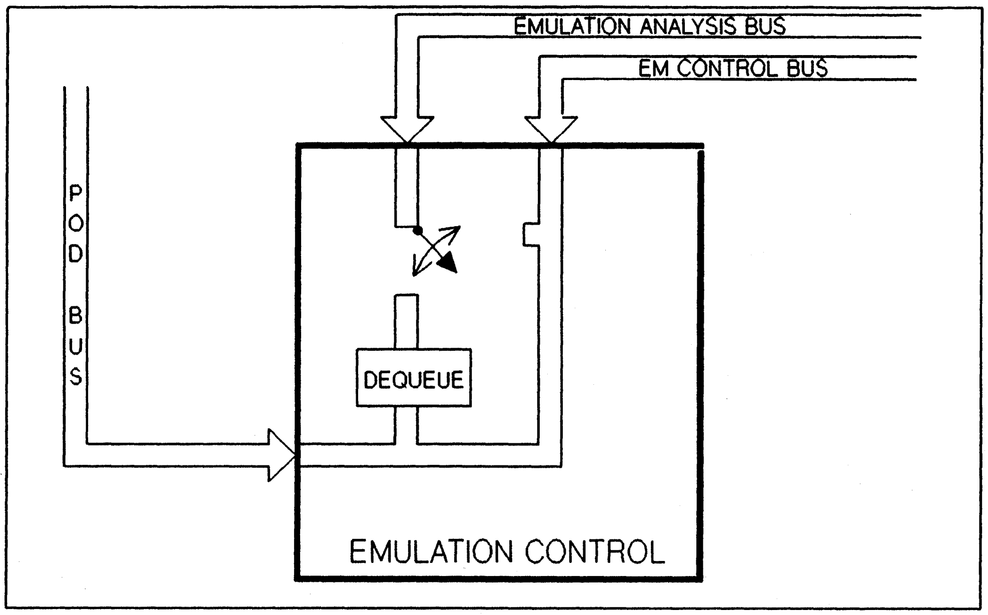
emulator probe:	contains the emulation processor, buffers, and control circuitry.
emulation controller:	converts the processor timing signals to standard 64000-UX timing signals.
memory controller:	controls the dual port memory shared by emulation and the 64000-UX system. Also contains the system address mapper.
emulation memory:	consists of the memory available to the user during emulation.
internal analysis:	captures address, data, and status information. Can also be used to cause a breakpoint on the occurrence of a specified trigger.
target system:	represents the system under design which probably contains a processor, control circuitry, memory, and I/O circuits.

STUDENT NOTES:

□ Introduction to Emulation

EMULATION

Split Emulation Bus Architecture (68000)

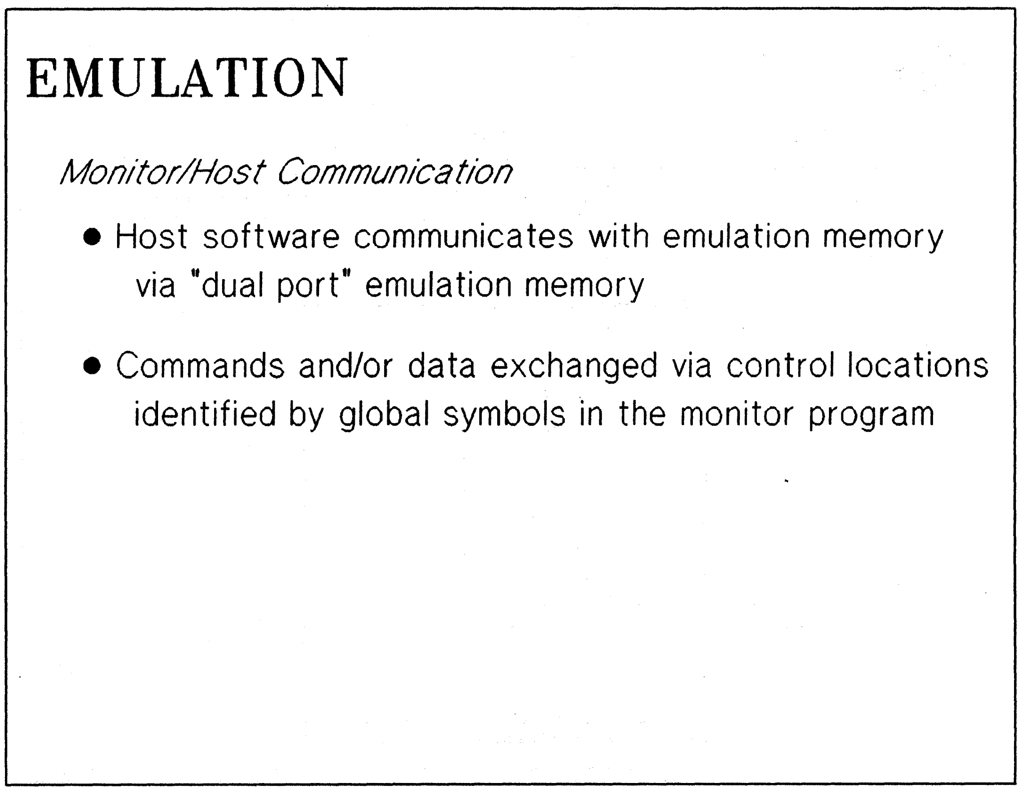


EM1 2.60 © 1987 Hewlett-Packard Company

NOTES:

Pre-fetching of instructions makes analysis more difficult. The "split bus" architecture allows only the valid (executed) information to be seen by the analyzer. The emulator usually also has a feature which allows the user to decide whether the pre-fetch instructions should be stored or filtered by the analyzer.

STUDENT NOTES:



EMULATION

Monitor/Host Communication

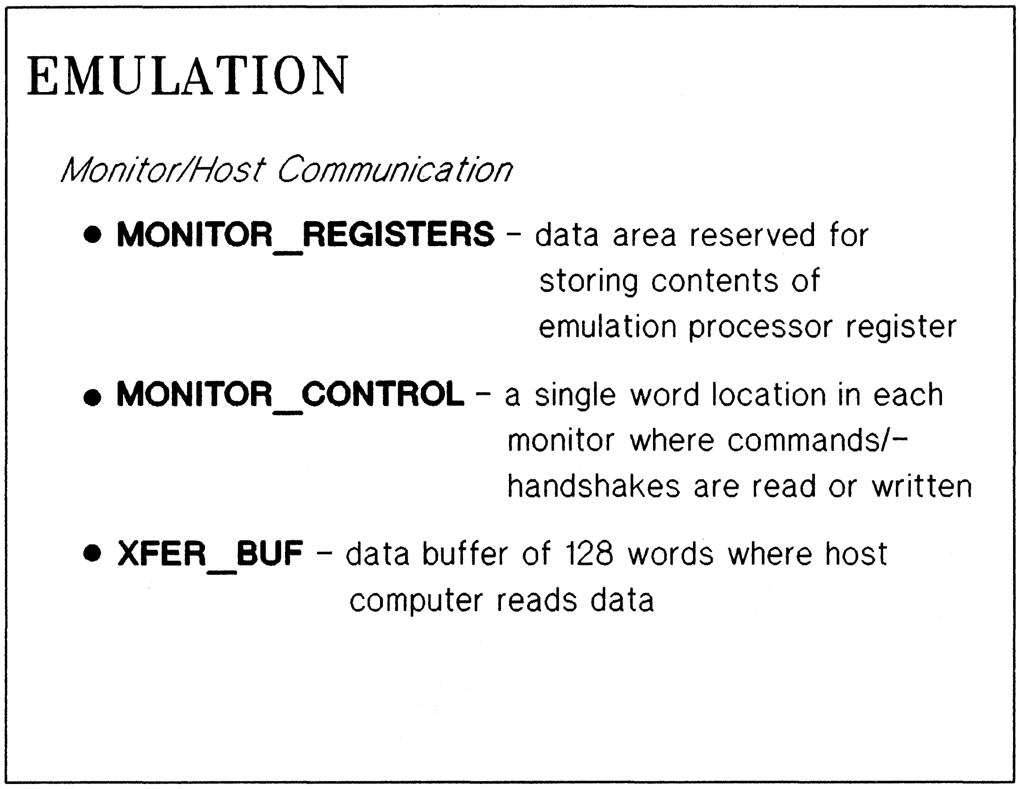
- Host software communicates with emulation memory via "dual port" emulation memory
- Commands and/or data exchanged via control locations identified by global symbols in the monitor program

EM1 2.65

NOTES:

The host processor communicates with the emulation processor by transferring data to and from emulation memory. Data transfer is accomplished through the memory control board into the static RAM board. There are three global symbols (except 8051) that identify control locations to exchange commands or data. The following slide will illustrate these symbols.

STUDENT NOTES:



EMULATION

Monitor/Host Communication

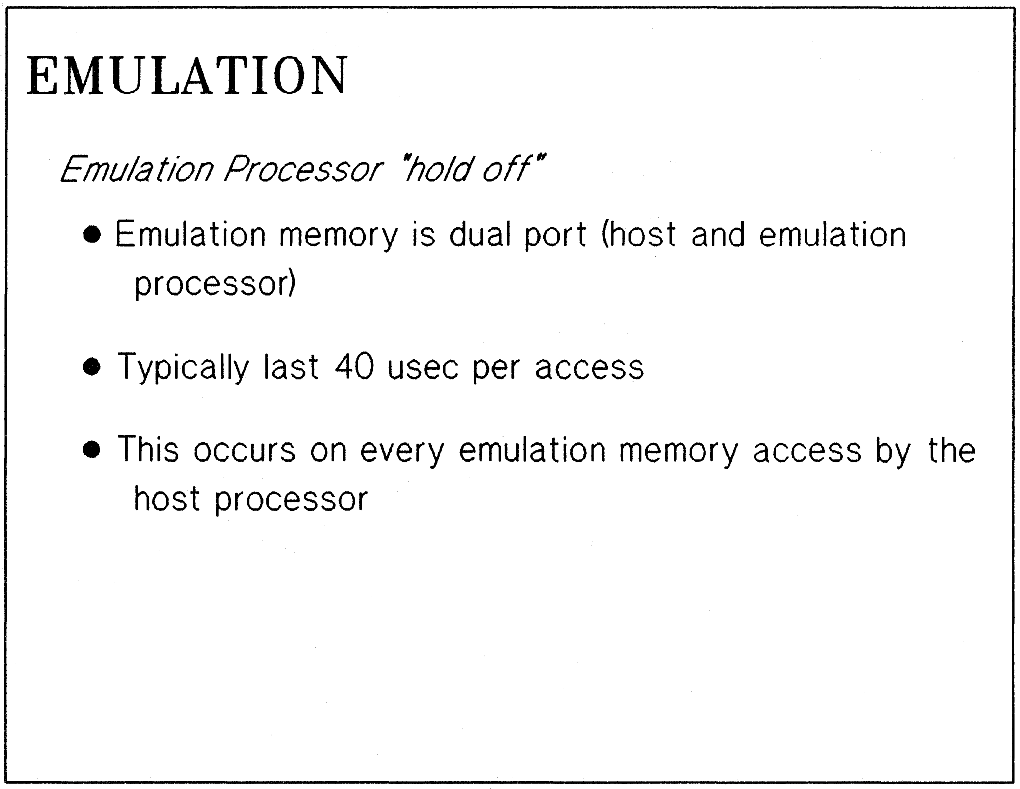
- **MONITOR_REGISTERS** - data area reserved for storing contents of emulation processor register
- **MONITOR_CONTROL** - a single word location in each monitor where commands/-handshakes are read or written
- **XFER_BUF** - data buffer of 128 words where host computer reads data

EM1 2.70 © 1987 Hewlett-Packard Company

NOTES:

Processor data register storage is located at MONITOR_REGISTERS. Command and data transfers are controlled by the symbols, MONITOR_CONTROL and XFER_BUF. Commands from the host processor are written into MONITOR_CONTROL. XFER_BUF is the location of the RAM buffer where the host exchanges data. Unless all three symbols are found, the emulation monitor will not be recognized.

STUDENT NOTES:



EMULATION

Emulation Processor "hold off"

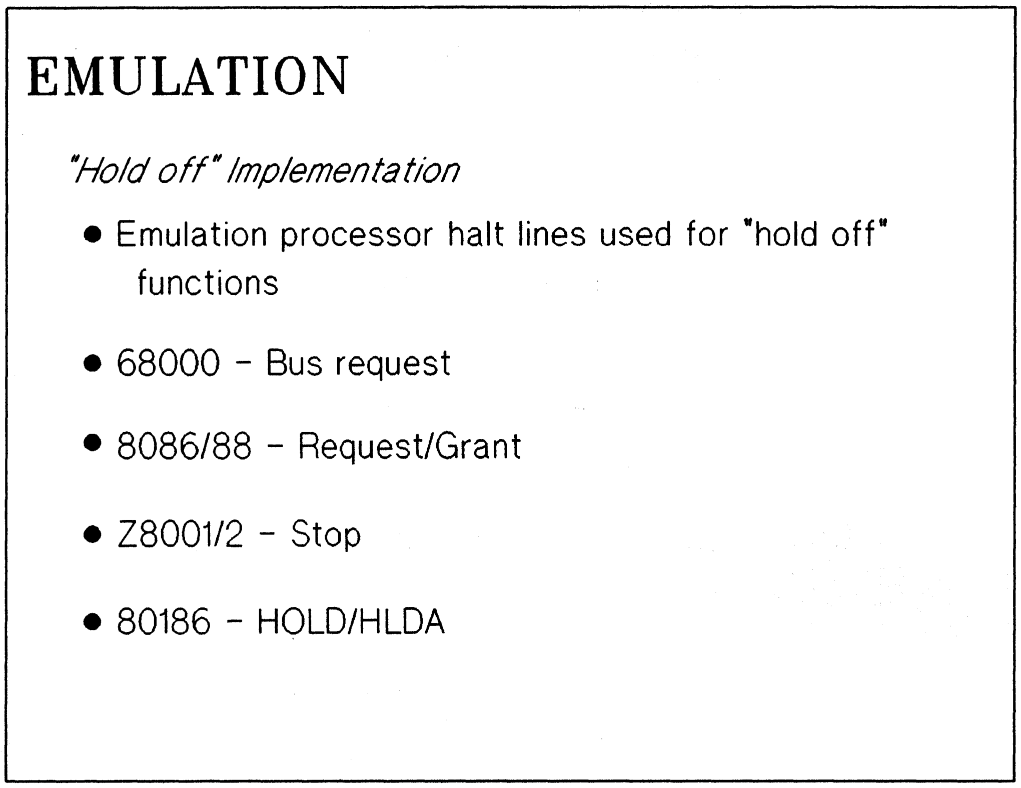
- Emulation memory is dual port (host and emulation processor)
- Typically last 40 usec per access
- This occurs on every emulation memory access by the host processor

EM1 2.75

NOTES:

Since the time between memory cycles in most emulators is very short, emulation memory cannot be used as a dual-port memory. Thus the emulation processor must be "held off" to access emulation memory. The user's target system sees nothing but what appears to be internal bus cycles. The "hold off" is usually implemented using the DMA control lines. The following slide gives some specific examples of what control lines certain processors use to implement this.

STUDENT NOTES:



EMULATION

"Hold off" Implementation

- Emulation processor halt lines used for "hold off" functions
- 68000 - Bus request
- 8086/88 - Request/Grant
- Z8001/2 - Stop
- 80186 - HOLD/HLDA

EM1 2.80 © 1987 Hewlett-Packard Company

STUDENT NOTES:

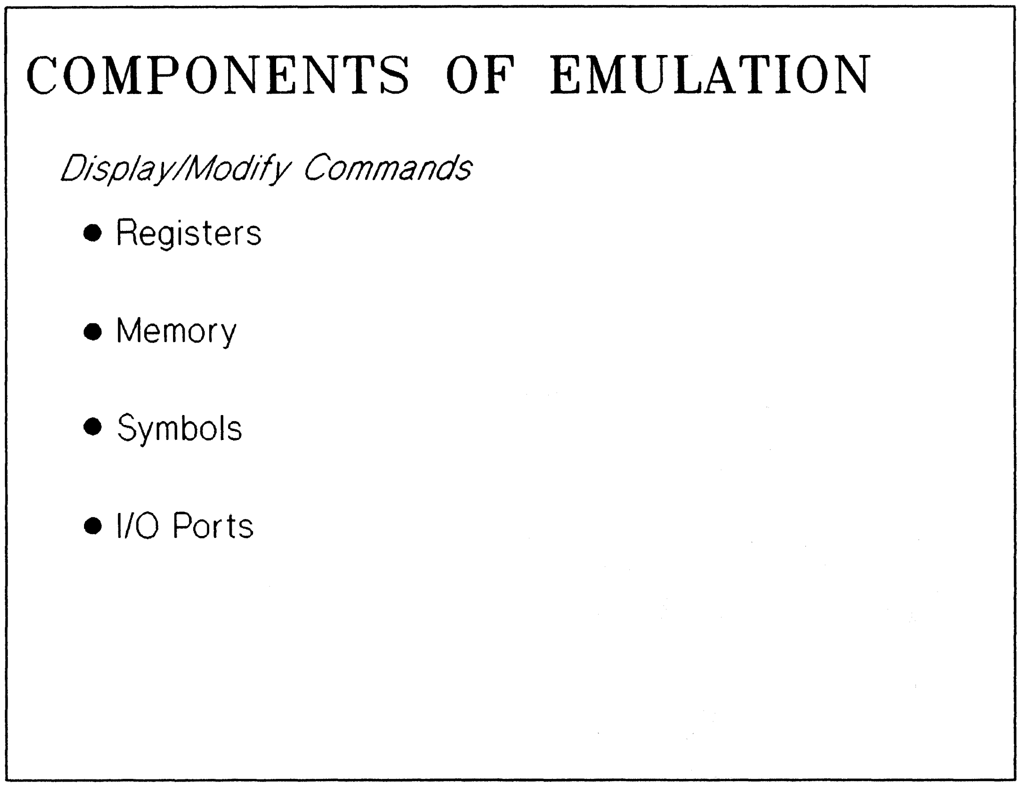
EMULATION

Components of Emulation

- Configuring the emulator
- Program loading and execution
- edbuild
- Run control
- ➡ Display/Modify commands
- Internal analysis with emulator
- Simulated I/O

EM1 2.83

STUDENT NOTES:



COMPONENTS OF EMULATION

Display/Modify Commands

- Registers
- Memory
- Symbols
- I/O Ports

EM1 2.85 © 1987 Hewlett-Packard Company

NOTES:

display registers: (page 4-16) this command is usually used in conjunction with the single step command. This allows the user to view the most minute changes in the registers from one instruction to the next.

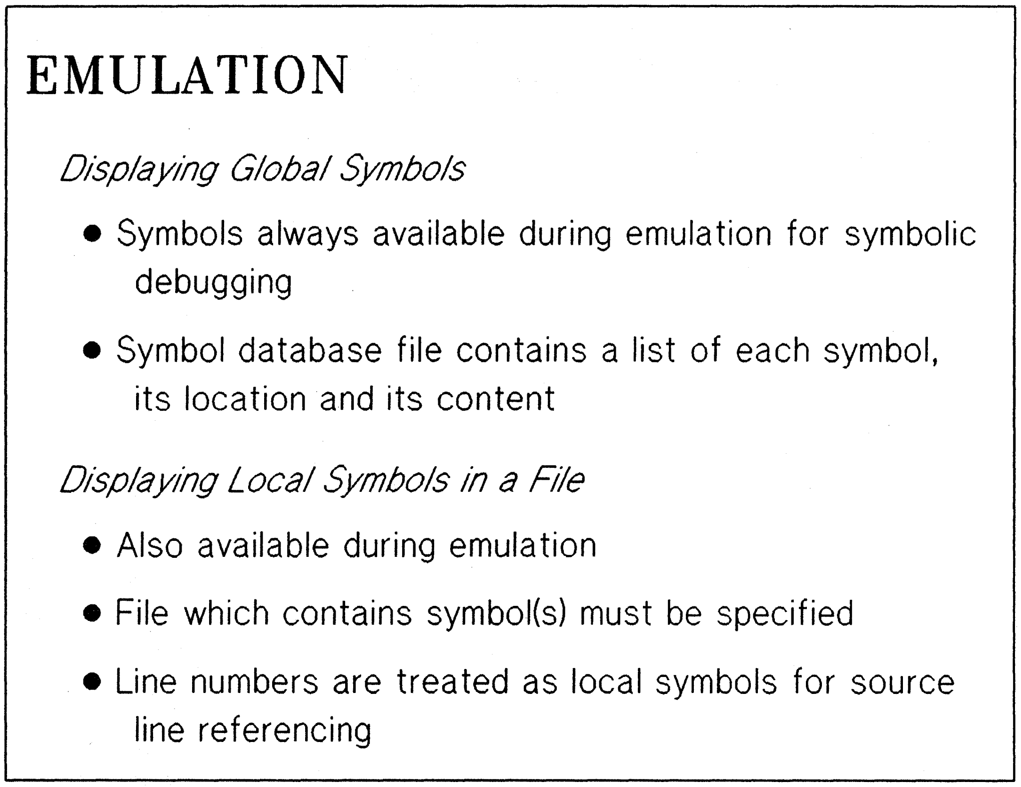
display / modify memory: (page 4-15) these commands give the user the flexibility to view and/or modify any addressable memory location(s). This feature would allow the user to modify I/O port control registers without having to write and assemble the otherwise necessary source code.

display local / global symbols: the user can display the local and global symbols of his code to determine their absolute addresses. The global symbols can then be used is setting up measurement specifications (symbolic debugging).

display / modify I/O ports: this can be helpful when checking the status of the target system or when setting up the control hardware of the target system.

STUDENT NOTES:

Introduction to Emulation



EMULATION

Displaying Global Symbols

- Symbols always available during emulation for symbolic debugging
- Symbol database file contains a list of each symbol, its location and its content

Displaying Local Symbols in a File

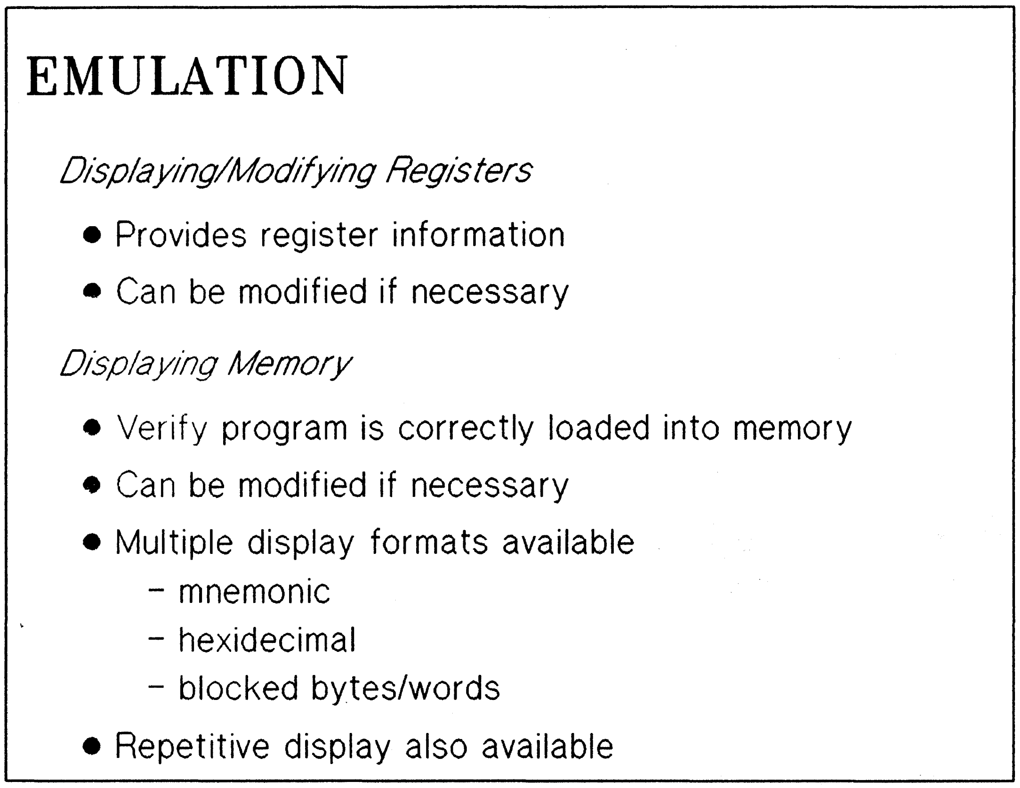
- Also available during emulation
- File which contains symbol(s) must be specified
- Line numbers are treated as local symbols for source line referencing

EMI 2.90

NOTES:

The ability to display global and local symbols gives 64000-UX one of its many features: symbolic debug. The actual symbol name can be used instead of a specific address. Another feature called "source line referencing" is possible considering the assembler/compiler uses each line number as a local symbol.

STUDENT NOTES:



EMULATION

Displaying/Modifying Registers

- Provides register information
- Can be modified if necessary

Displaying Memory

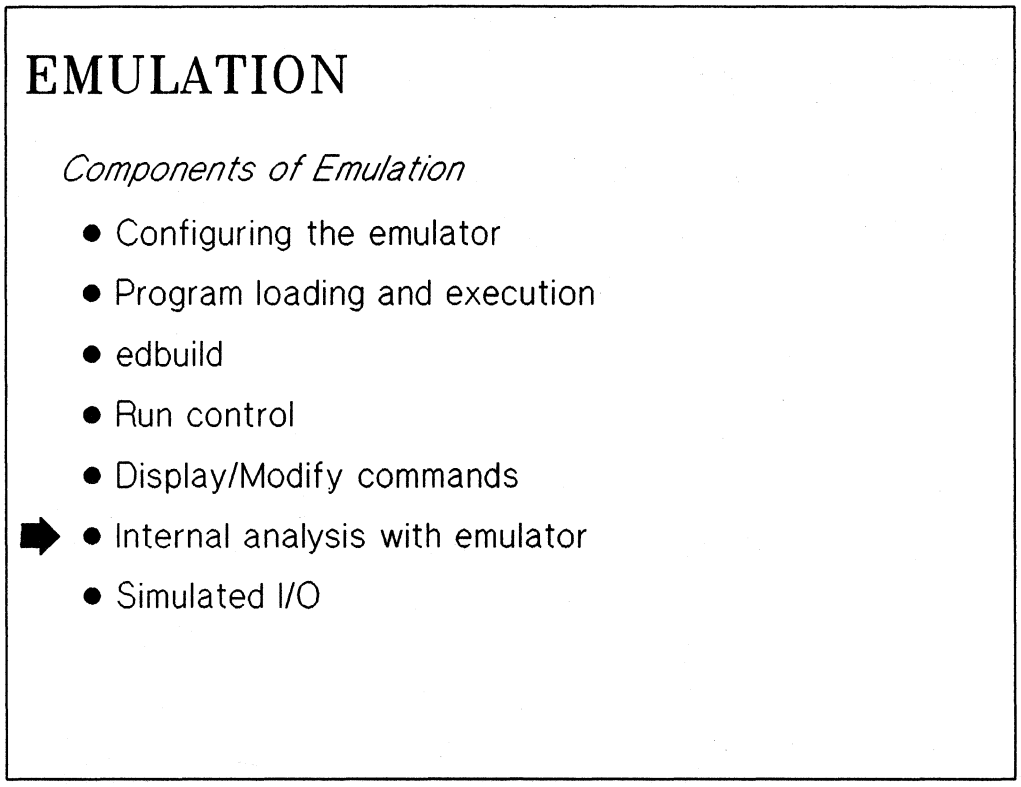
- Verify program is correctly loaded into memory
- Can be modified if necessary
- Multiple display formats available
  - mnemonic
  - hexidecimal
  - blocked bytes/words
- Repetitive display also available

EM1 2.95

NOTES:

With the ability to specify a range when using some of these commands, the user can modify entire sections of memory with one easy command.

STUDENT NOTES:



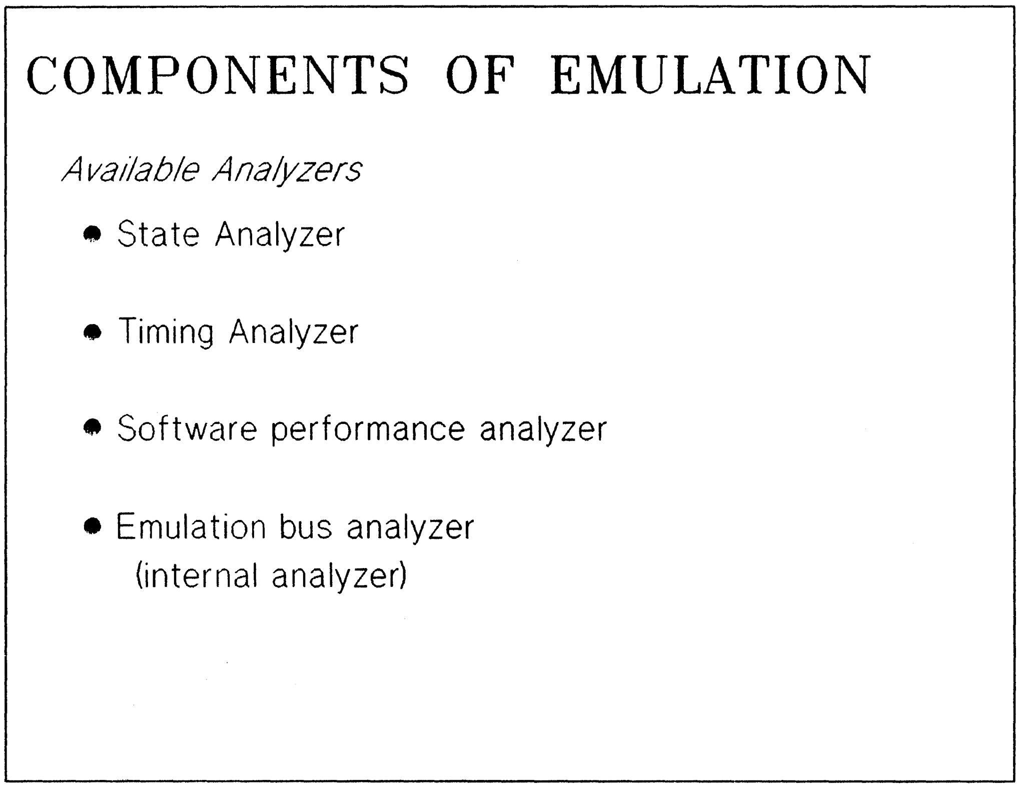
EMULATION

Components of Emulation

- Configuring the emulator
- Program loading and execution
- edbuild
- Run control
- Display/Modify commands
- ➡ Internal analysis with emulator
- Simulated I/O

EM1 2.98

STUDENT NOTES:



COMPONENTS OF EMULATION

Available Analyzers

- State Analyzer
- Timing Analyzer
- Software performance analyzer
- Emulation bus analyzer
  (internal analyzer)

EM1 3.00 © 1987 Hewlett-Packard Company

MANUAL: pages 5-1 thru 5-10 8 and 16 Bit Emulation Reference

NOTES:

state analyzer (64620): this analyzer is used as a software debugging tool, utilizing its extensive trace / displaying capabilities. The state analyzer has up to 120 10MHz data acquisition channels and up to 8 10MHz clock acquisition channels.

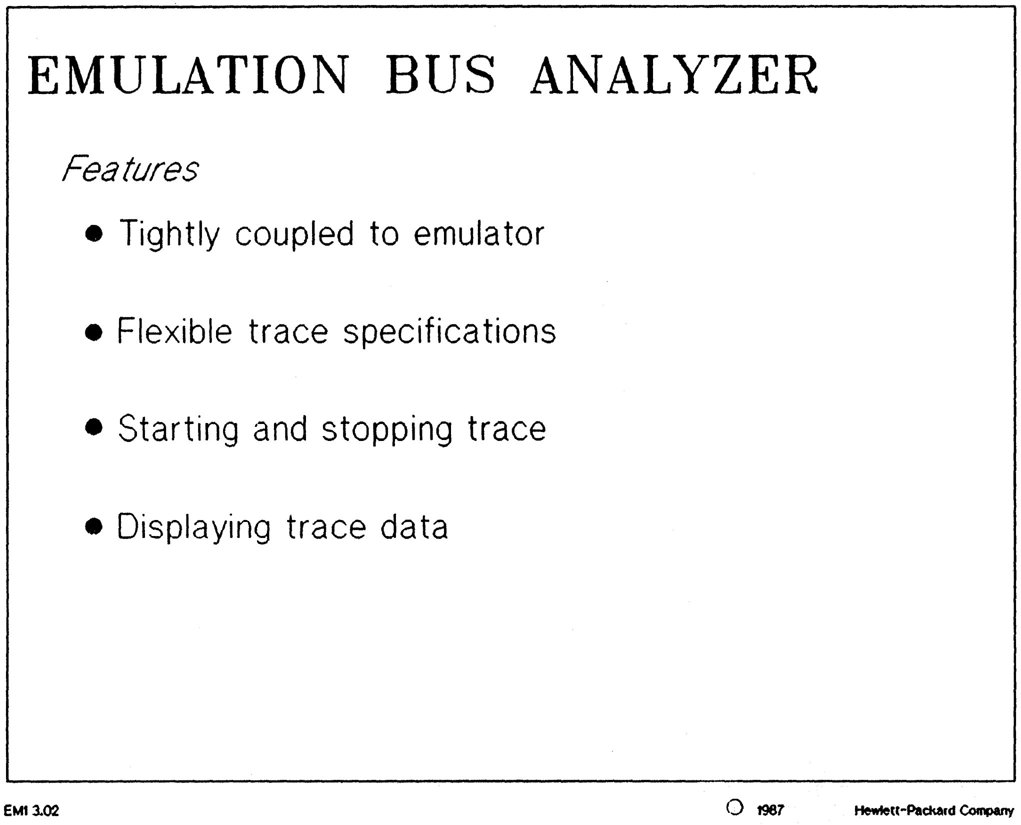
timing analyzer (64610): this analyzer is used to debug hardware problems associated with target system timing. The timing analyzer has up to 32 200MHz timing acquisition channels.

software performance analyzer (64310): used to measure program activity and module duration.

NOTE: these analyzers are beyond the scope of this course. Advanced training and SE (Systems Engineer) consulting is currently available for all these products.

internal analyzer (64302): used in conjunction with an emulator to provide productive emulation bus analysis. The functionality of this analyzer will be covered in some detail in this course.

STUDENT NOTES:



EMULATION BUS ANALYZER

Features

- Tightly coupled to emulator
- Flexible trace specifications
- Starting and stopping trace
- Displaying trace data

EMI 3.02 © 1987 Hewlett-Packard Company

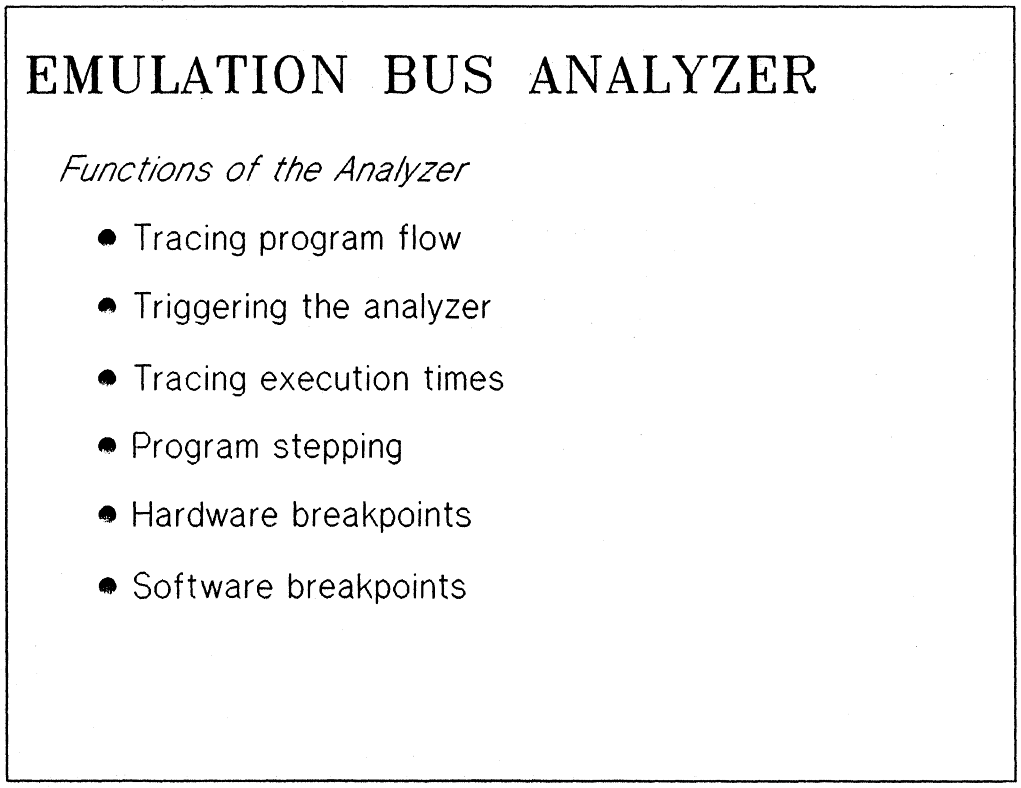
NOTES:

The internal analyzer board is the most tightly coupled board to an emulation system because it functions as the IMB source for the emulator.

The internal analyzer gives a flexible set of trigger conditions which can be combined to produce efficient and effective measurements. Some of these features include:

trace after / about / before / only commands
address / data / status triggers
don't care (x) specifications
OR and AND operators in trace specifications
absolute and relative time counters
multiple formats for displaying trace data
IMB connection to other 64000-UX analyzers

STUDENT NOTES:



EMULATION BUS ANALYZER

Functions of the Analyzer

- Tracing program flow
- Triggering the analyzer
- Tracing execution times
- Program stepping
- Hardware breakpoints
- Software breakpoints

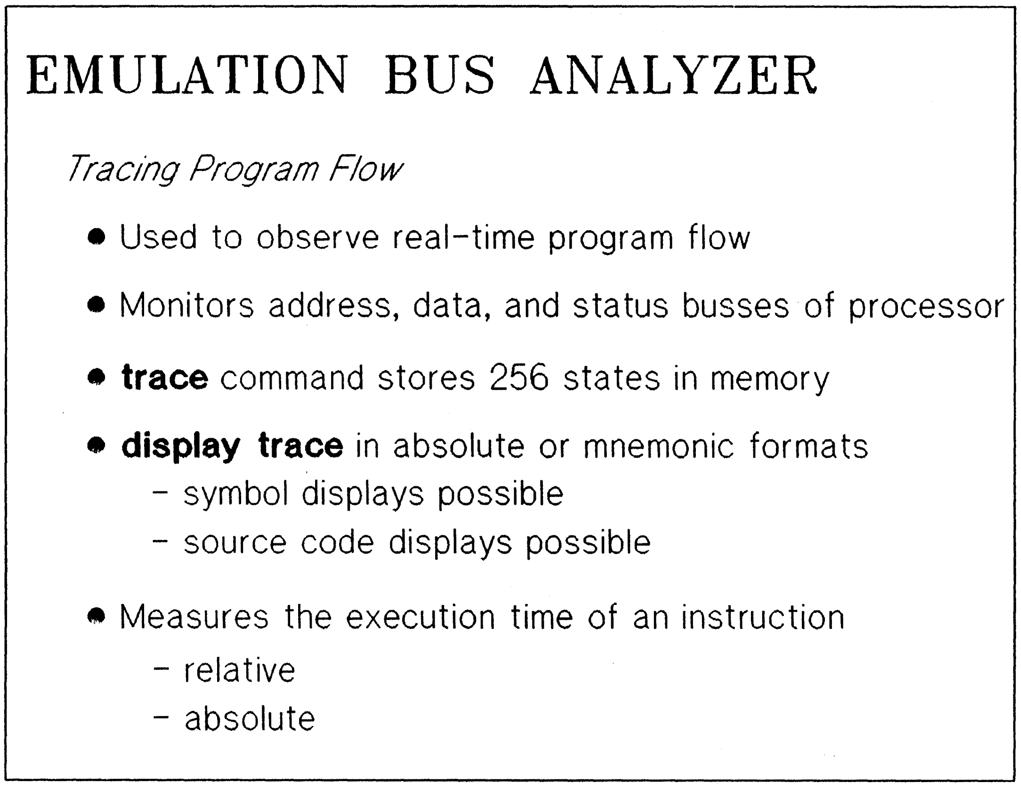
EM1 3.05

NOTES:

The internal analyzer is used to monitor the address, data, and status busses of the emulator in real-time.

The internal analyzer has many nice features listed above which can be used in combination with each other to make the analyzer an even more powerful tool.

STUDENT NOTES:



EMULATION BUS ANALYZER

Tracing Program Flow

- Used to observe real-time program flow
- Monitors address, data, and status busses of processor
- **trace** command stores 256 states in memory
- **display trace** in absolute or mnemonic formats
  - symbol displays possible
  - source code displays possible
- Measures the execution time of an instruction
  - relative
  - absolute

EM1 3.10

NOTES:

Two of the nicest features of the internal analyzer are the ability to display data in multiple formats and the time stamping of user code. The execution time counter even has the ability to display execution times of user's code in relative or absolute format.

STUDENT NOTES:

EMULATION BUS ANALYZER

Triggering the Analyzer

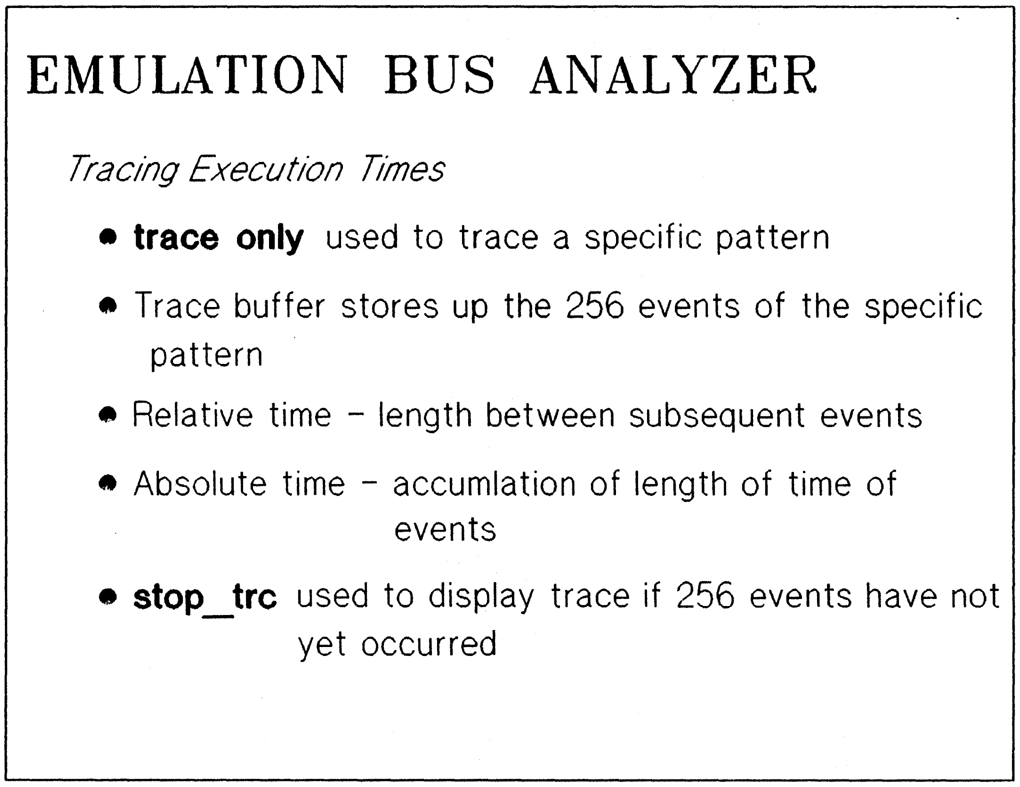
- Trigger condition
  - address pattern
  - data pattern
  - status pattern
  - can be "ORed" for up to two trigger conditions
- Trigger position is flexible
  - beginning - **trace after**
  - middle - **trace about**
  - end - **trace before**
- Trigger buffer
  - stores up to 256 data sets

EMI 3.15 ◯ 1987 Hewlett-Packard Company

NOTES:

The ability to flexibly trigger the internal analyzer makes it a very powerful debugging tool. Depending on what data the user wishes to capture, the analyzer is equipped to accept almost any combination of trigger condition and trigger position. Many of these features will be learned about in the lab.

STUDENT NOTES:



EMULATION BUS ANALYZER

Tracing Execution Times

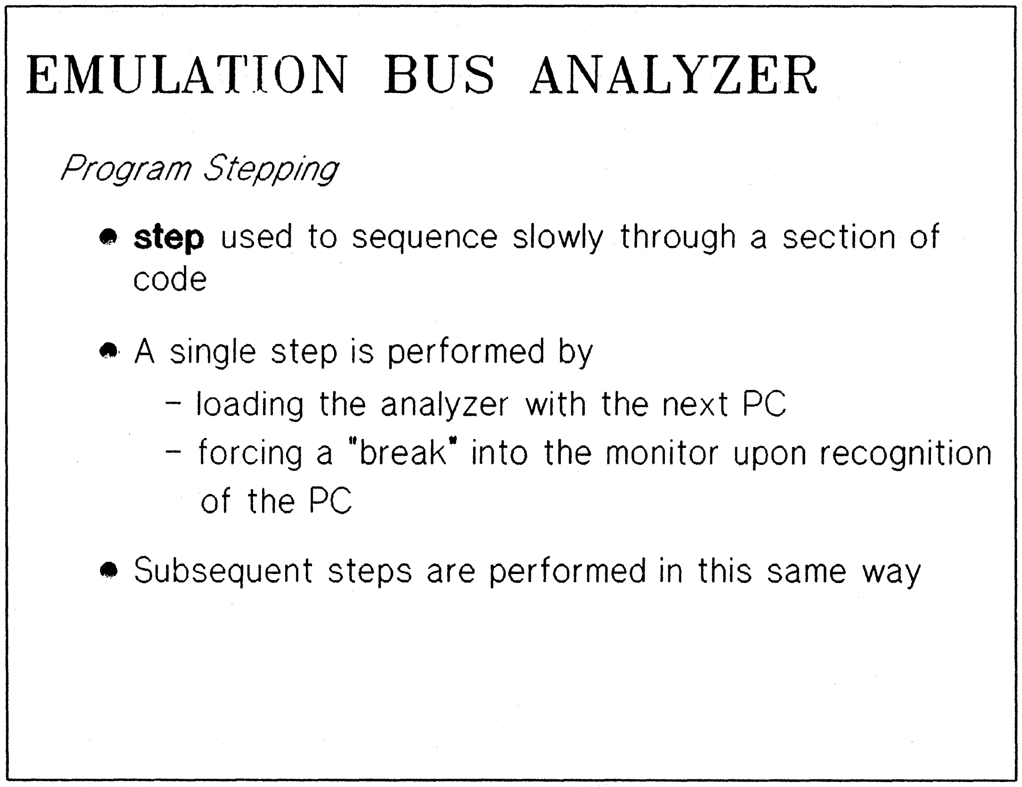
- **trace only** used to trace a specific pattern
- Trace buffer stores up the 256 events of the specific pattern
- Relative time - length between subsequent events
- Absolute time - accumlation of length of time of events
- **stop_trc** used to display trace if 256 events have not yet occurred

EM1 3.20 © 1987 Hewlett-Packard Company

NOTES:

The ability of the internal analyzer to trace execution times makes it a fantastic tool for testing timing loops in software.

STUDENT NOTES:



EMULATION BUS ANALYZER

Program Stepping

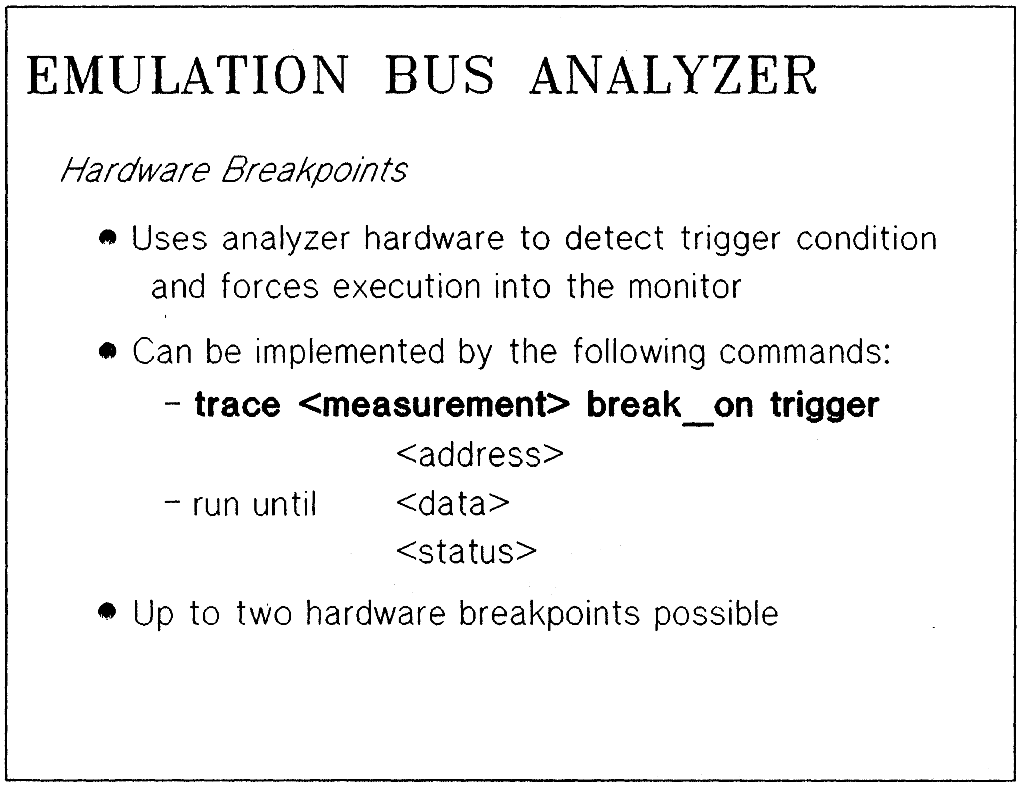
- **step** used to sequence slowly through a section of code
- A single step is performed by
  - loading the analyzer with the next PC
  - forcing a "break" into the monitor upon recognition of the PC
- Subsequent steps are performed in this same way

EMI 3.25 © 1987 Hewlett-Packard Company

NOTES:

The step instruction is another extremely important feature which would not be possible without the internal analyzer. The analyzer is used to set a trigger point to the next program counter (PC) and force a break into the monitor program when it detects the next PC.

STUDENT NOTES:



EMULATION BUS ANALYZER

Hardware Breakpoints

- Uses analyzer hardware to detect trigger condition and forces execution into the monitor
- Can be implemented by the following commands:
  - **trace <measurement> break_on trigger**

```
                 <address>
  - run until    <data>
                 <status>
```

- Up to two hardware breakpoints possible

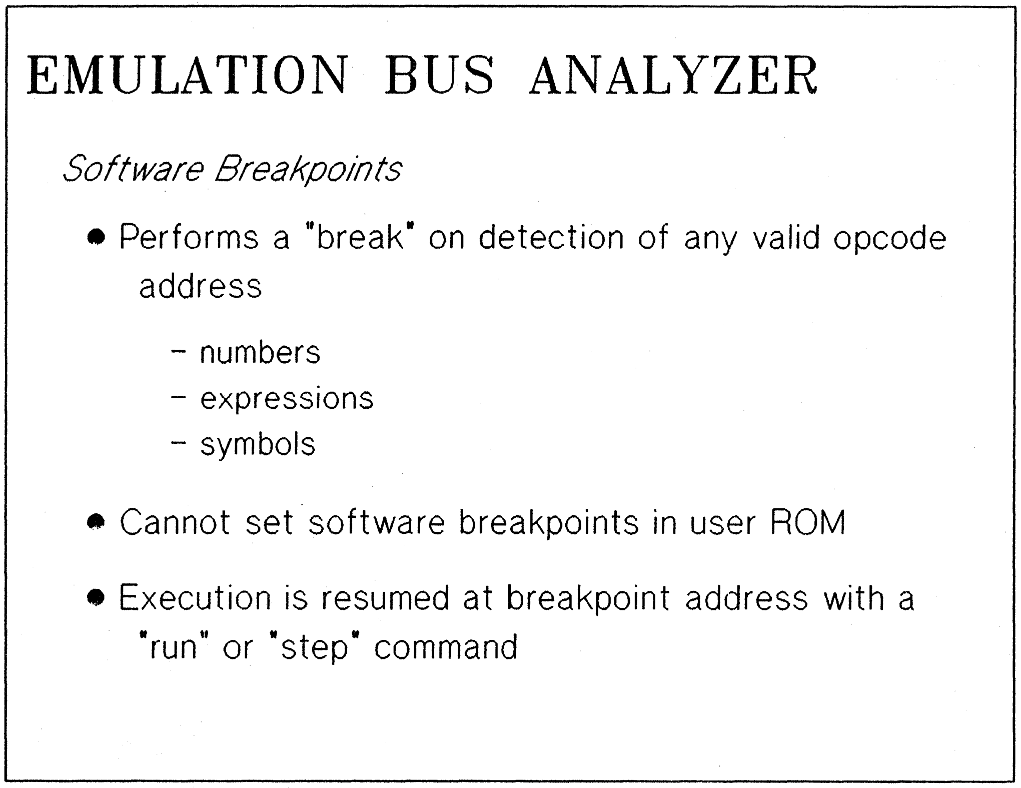
EMI 3.30 © 1987 Hewlett-Packard Company

NOTES:

A breakpoint is a predefined stopping point. A hardware breakpoint causes execution to be transferred to the monitor as soon as the detection of the trigger pattern occurs.

The "run until" command performs the same function as if the user had explicitly specified to break in to the background after meeting a trigger condition.

STUDENT NOTES:



EMULATION BUS ANALYZER

Software Breakpoints

- Performs a "break" on detection of any valid opcode address
  - numbers
  - expressions
  - symbols
- Cannot set software breakpoints in user ROM
- Execution is resumed at breakpoint address with a "run" or "step" command

EM1 3.35 © 1987 Hewlett-Packard Company

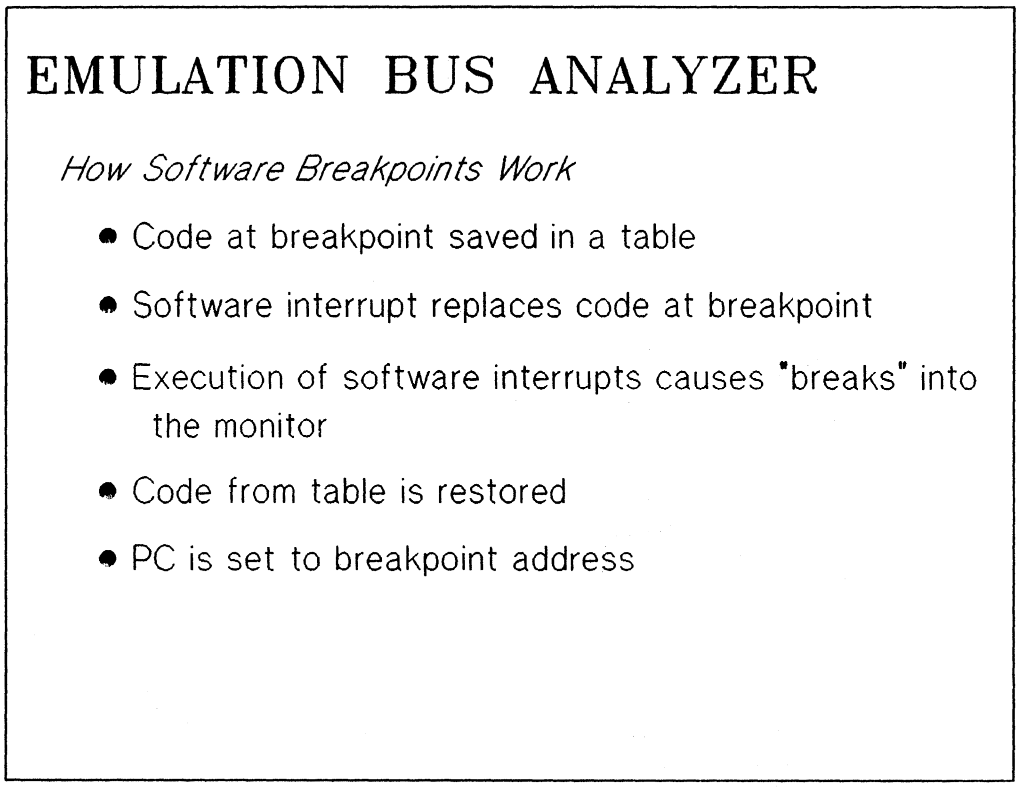
MANUAL: pages 4-11 thru 4-13 68000 Emulation w/Int. Analysis

NOTES:

Software breakpoints are an extremely useful feature, but not every 64000-UX emulator has the ability to use them. A software breakpoint is implemented in the user code as a software interrupt.

The software breakpoint must be set at an address which is the opcode of an instruction. If it was set in the middle of the data operand of an instruction, terrible things might happen.

STUDENT NOTES:



EMULATION BUS ANALYZER

How Software Breakpoints Work

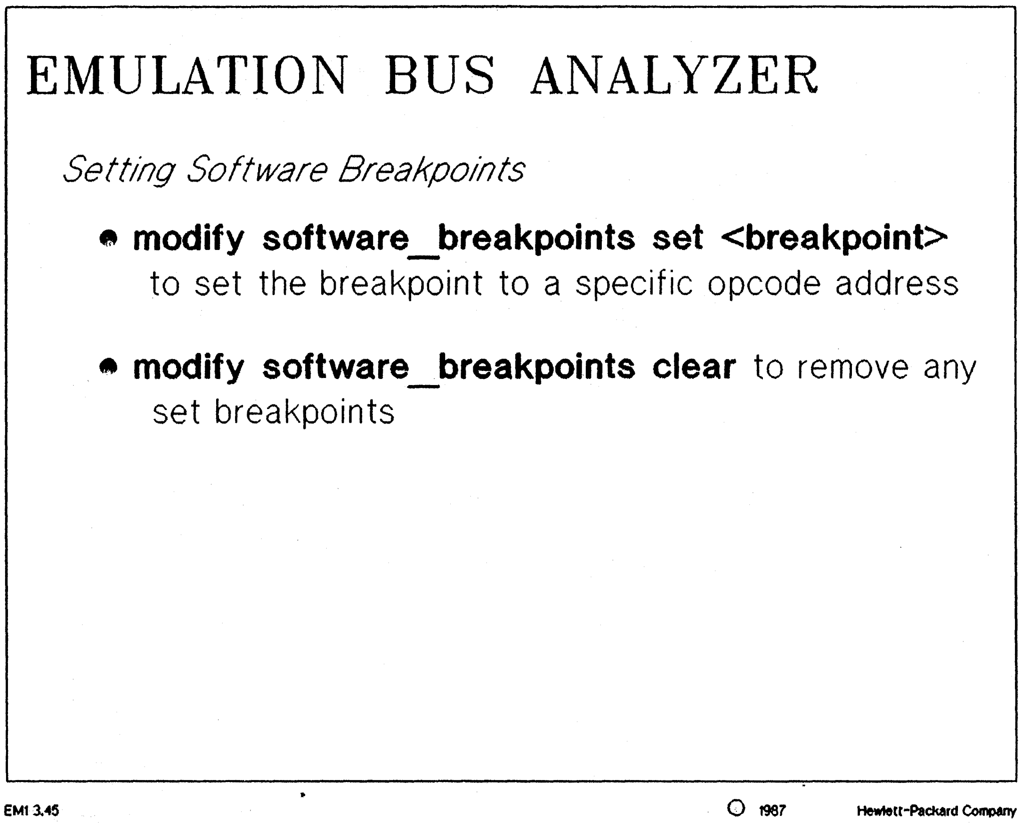
- Code at breakpoint saved in a table
- Software interrupt replaces code at breakpoint
- Execution of software interrupts causes "breaks" into the monitor
- Code from table is restored
- PC is set to breakpoint address

EM1 3.40 © 1987 Hewlett-Packard Company

NOTES:

The implementation of a software breakpoint is easy to understand. The user specifies a location that he wishes to be the software breakpoint (on an opcode). The real opcode at the location is stored in a table and replaced with a software interrupt opcode. When the user executes the code and hits a software interrupt, a break is caused and control is transferred to the monitor. The real opcode is then replaced in its original location and the PC is decremented accordingly.

STUDENT NOTES:



EMULATION BUS ANALYZER

Setting Software Breakpoints

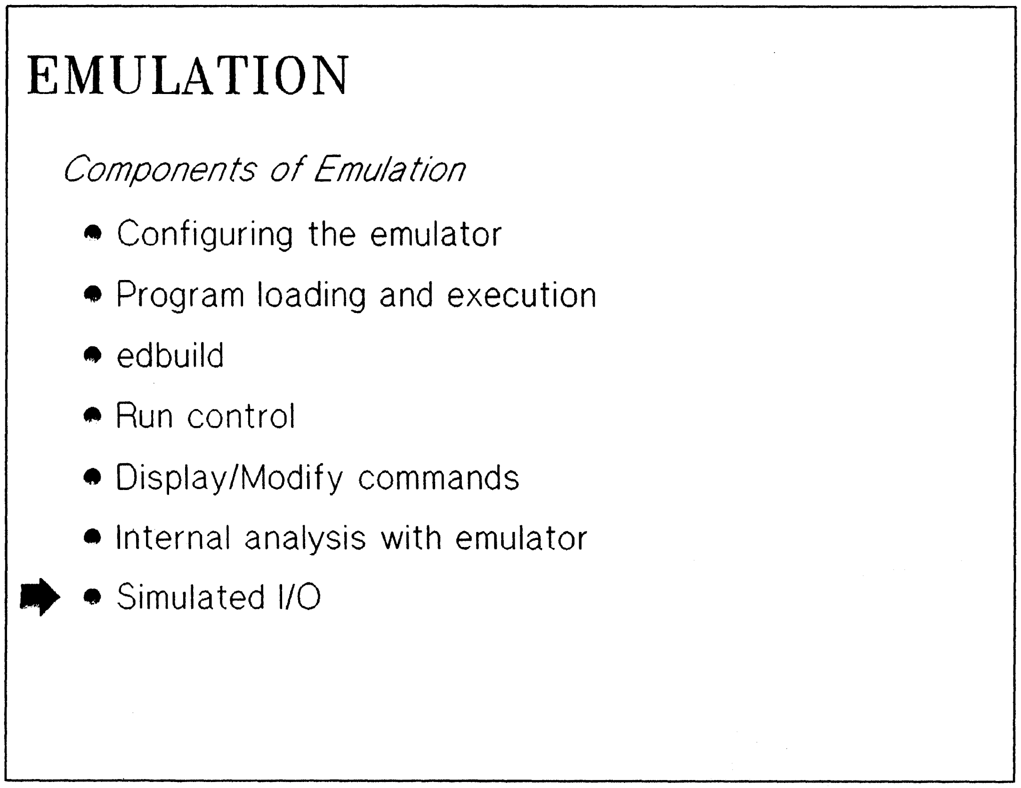
- **modify software_breakpoints set <breakpoint>** to set the breakpoint to a specific opcode address
- **modify software_breakpoints clear** to remove any set breakpoints

EMI 3.45 © 1987 Hewlett-Packard Company

NOTES:

Setting up and modifying software breakpoints can get to be an involved process. For the 68000, a specific trap address has to be picked and commented out of the monitor. The process is simple from the stand point of "modifying" the breakpoint to a specific address or removing a software breakpoint. Once again, this is a very powerful feature of the 64000-UX program development environment.

STUDENT NOTES:



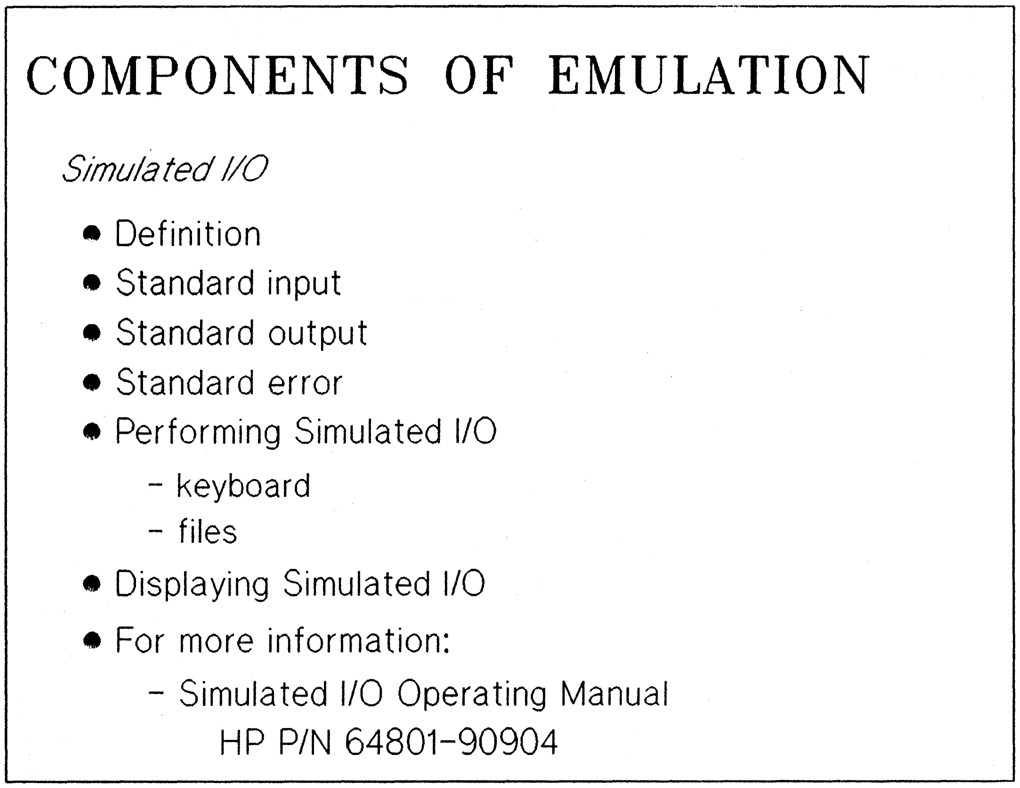
EMULATION

Components of Emulation

- Configuring the emulator
- Program loading and execution
- edbuild
- Run control
- Display/Modify commands
- Internal analysis with emulator
- ➡ Simulated I/O

EMI 3.48

STUDENT NOTES:



COMPONENTS OF EMULATION

Simulated I/O

- Definition
- Standard input
- Standard output
- Standard error
- Performing Simulated I/O
  - keyboard
  - files
- Displaying Simulated I/O
- For more information:
  - Simulated I/O Operating Manual
    HP P/N 64801-90904

EMI 3.50 © 1987 Hewlett-Packard Company

NOTES:

simulated I/O: a process which allows your emulation system to communicate with HP 9000 files and your workstation keyboard and display. Simulated I/O also allows your emulation system to execute HP-UX commands, and thus can communicate with other 9000 series 300 devices, such as printers, plotters, modems, etc.

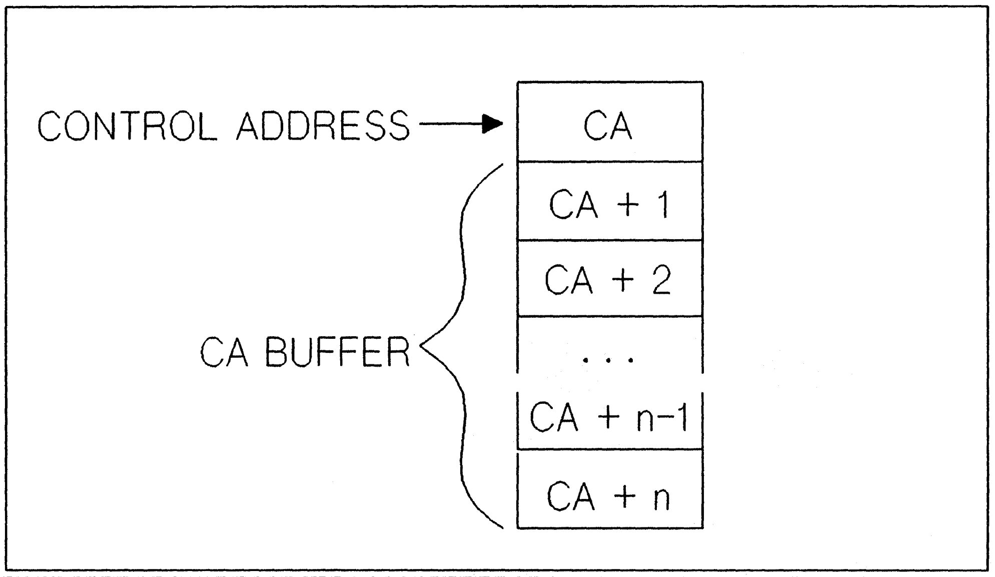
Simulated I/O can be tied into and redirect the standard output, standard error, and standard input devices of the 9000 series 300. This allows the user to effectively redirect the input and output of his target system program to the keyboard and display of the 9000 series 300.

This topic is beyond the scope of this course. Further training on simulated I/O can be obtained by taking an advanced emulation course.

STUDENT NOTES:

SIMULATED I/O

Emulation System Request



EM1 3.55 © 1987 Hewlett-Packard Company

NOTES:

Communications between your emulation system and HP 9000 files takes place through contiguous byte length emulation system memory locations. The first memory location is called the Control Address (CA). The Control Address and the memory locations which follow it are referred to as the CA buffer.

The Control Address (CA) buffer should be located in emulation RAM. It is possible, however, for the CA buffer to be located in target system RAM, but there will be a performance penalty for doing so.

Emulation System Request:

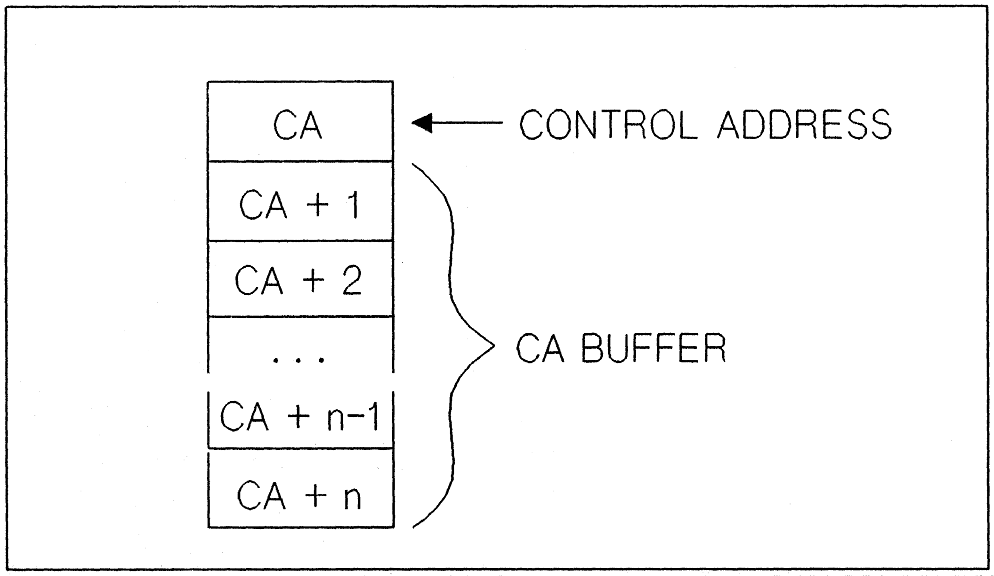
(1) Simulated I/O command parameters are first placed into the appropriate CA buffer locations.

(2) After the CA buffer is loaded with simulated I/O command parameters, the command code is placed into the Control Address (CA) to cause the execution of the command.

STUDENT NOTES:

SIMULATED I/O

HP 64000-UX Response



EM1 3.60 © 1987 Hewlett-Packard Company

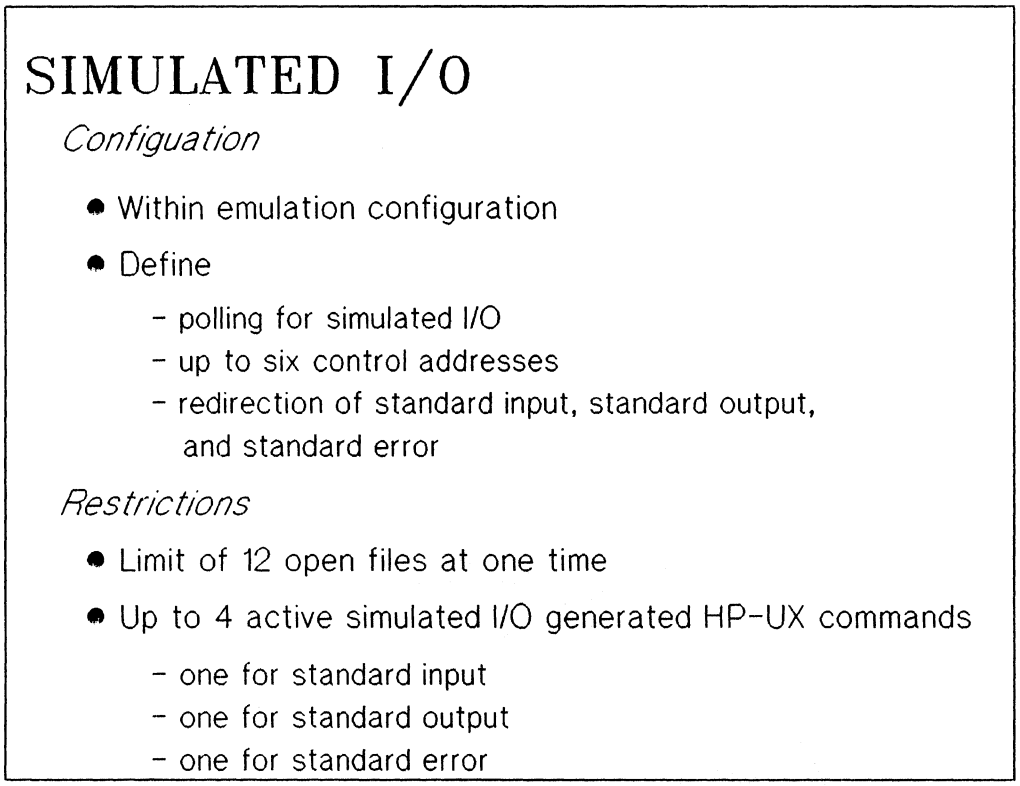
NOTES:

HP 64000-UX Response:

(1) Before returning a value to the Control Address, some simulated I/O operations return additional information to the CA buffer.

(2) If the operation was successful, a 00H is returned to the Control Address. If the operation was not successful, an error code is returned to the CA.

STUDENT NOTES:



SIMULATED I/O

Configuation

- Within emulation configuration
- Define
  - polling for simulated I/O
  - up to six control addresses
  - redirection of standard input, standard output, and standard error

Restrictions

- Limit of 12 open files at one time
- Up to 4 active simulated I/O generated HP-UX commands
  - one for standard input
  - one for standard output
  - one for standard error

EMI 3.65

© 1987 Hewlett-Packard Company

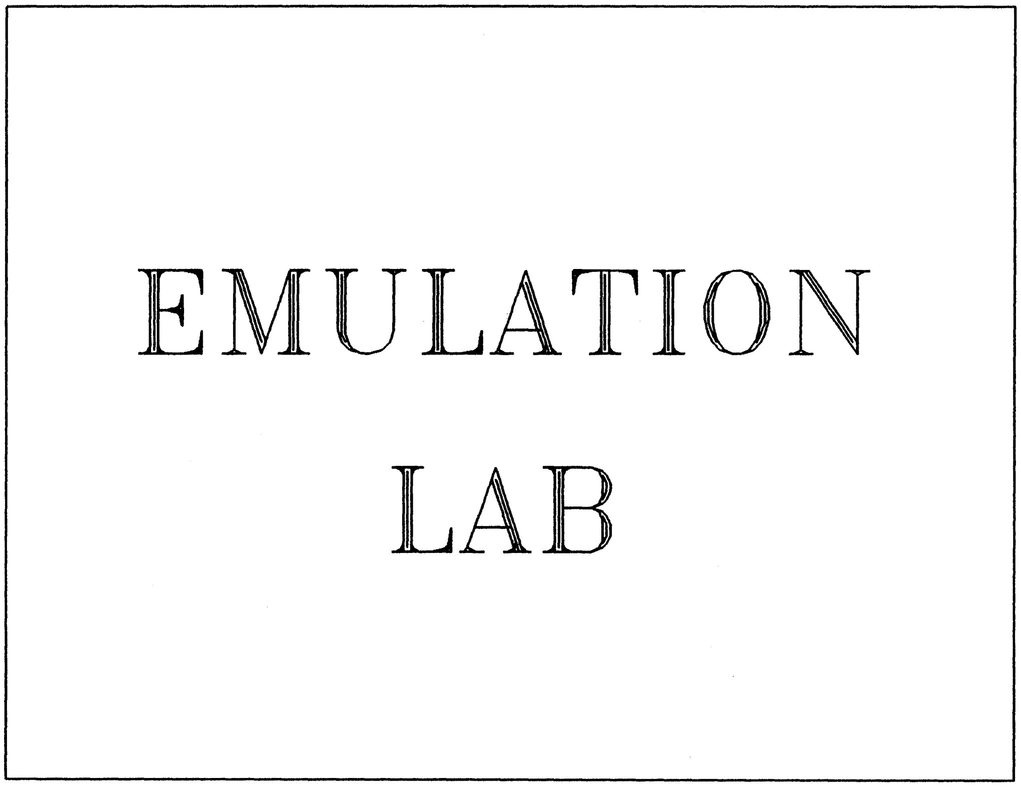
NOTES:

The simulated I/O subsystem must be set up by answering a series of configuration questions. These questions are a part of the general emulation configuration. They deal with enabling simulated I/O, setting the control address(es), and defining files used for standard I/O.

If each of the maximum of four simultaneous simulated I/O processes has files for standard input, standard output, and standard error; then the maximum of 12 files open simultaneously is understandable.

STUDENT NOTES:

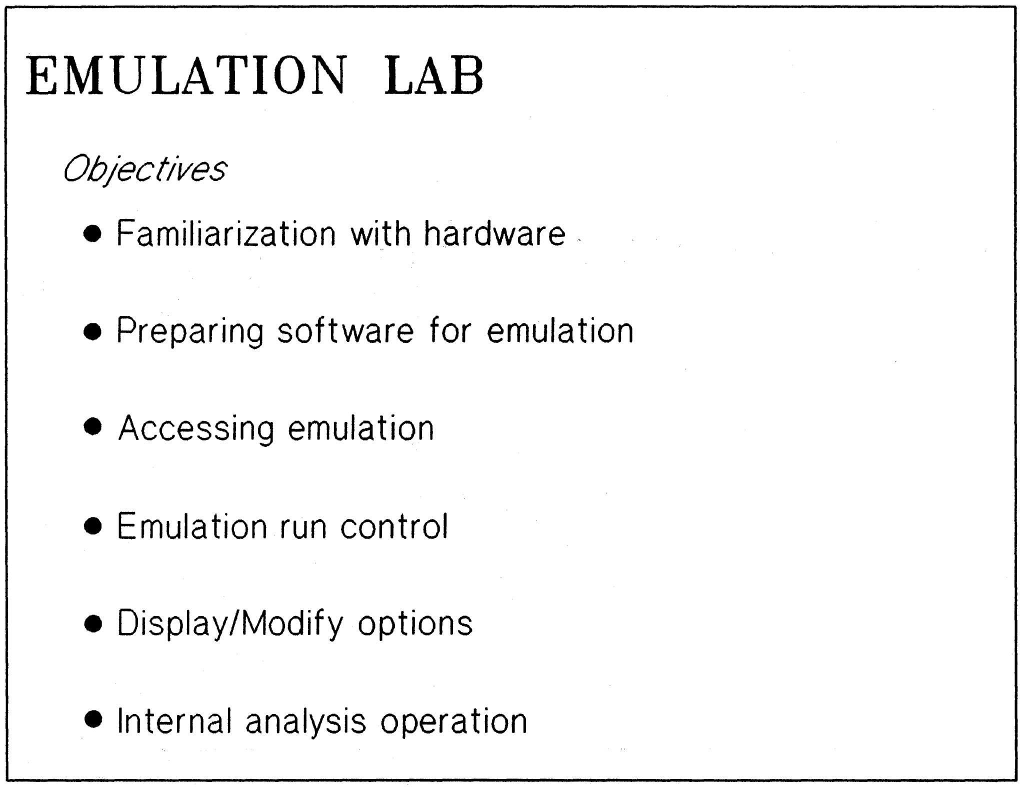
□ Emulation Lab



EMULATION

LAB

EM1 3.70



EMULATION LAB

Objectives

- Familiarization with hardware
- Preparing software for emulation
- Accessing emulation
- Emulation run control
- Display/Modify options
- Internal analysis operation

EM1 3.75

STUDENT NOTES:

HP 64000-UX EMULATION LAB

Objectives:

* prepare the absolute code for emulation
* accessing the emulator
* emulation run control
* display / modify options
* internal analysis operation

The program to be used with this lab is a game called "Towers of Hanoi". This program is written in C for the most part, but does include a Pascal "pause" routine to show the flexibility of the language system. Another concept which will be introduced is the idea of an emulation monitor program. This program is written in 68000 assembly language and is needed to make the 68000 emulator function properly. It is user-modifiable source code and needs to be assembled and linked before using.

A brief description of each of the files used in the "Towers of Hanoi" program is shown below:

<filename>	<description>	<implemented language>
mon_68000.S	the 68000 emulator monitor program	(assembly)
main.C	the main routine of the Towers program	(C)
pause.P	pause routine used to slow down the output	(Pascal)
towers.C	recursive routine for moving discs to pegs	(C)
data	file for setting data parameters	(C)
data_comp.C	contains data structure for discs and pegs	(C)
clr_display.C	routine for clearing game display	(C)
assign_discs.C	initialization routine for Hanoi	(C)
gen_display.C	generate next display for game	(C)
move_disc.C	procedure to remove and place discs	(C)
place.C	places disc on a given peg	(C)
remove.C	remove a disc from a given peg	(C)
comp_hanoi	command file which assembles and links all modules	
all.K	a linker command file used to link the relocatables	

The structure of how the different routines are called is shown below:

```
                          /------- main.C ------\
                         /                       \
            gen_display.C                         towers.C
              /        \                             |
   clr_display.C    assign_discs.C               move_disc.C
                                                /     |      \
                                       remove.C   place.C   pause.P
   data_comp.C ---- loaded into memory
   data        ---- included with other modules
   mon_68000.S ---- loaded into memory as emulation monitor
```

☐ Emulation Lab

The student will need to know how to use an editor during this lab.
The Sk editor is probably the best choice since it is easier to use
and learn then the Vi editor.

Windowing is a neat feature of the 9000 series 300 and it is recommended
that you use the Windex (HP 9000 Windows) during the lab.
You start the Windex system by typing the "wmstart" command from an HP-UX
prompt; only on a medium or high resolution bit-mapped display.

The X Window system may also be available. Type the command "xstart"
at an HP-UX prompt; only on a medium or high resolution bit-mapped display.
Since the X Window system is an option for HP-UX, it may not be available
on your 9000 series 300 computer.

Manuals for both the windowing systems are available as part of the
training course. The use of either windowing package may degrade the
system performance. If you need help with windows, ask the instructor.

The lab will be broken into two major sections: the features of emulation
and the troubleshooting section.

Prepare absolute code

[ ] Enter pmon and change directories to the "$HOME/emul" directory.

[ ] A command file for compiling all of the necessary files has been
provided for you. Take a look at comp_hanoi and take notice of the
options that are used for compiling the source files.

What do each of the options do for you?

What is the .o file that is at the end of each line?

[ ] Use the linker command file supplied (all.K) to link the relocatables.
The absolute file will be named "main.X". You should generate a
linker listing file. Use the following command to do this:

```
lnk -ox all.K > main.map
```

☐ Emulation Lab

Accessing Emulation

[ ] Enter the measurement system software and setup a measurement system including only the 68000 emulator. Name the measurement system "<username>" and add the 68000 module naming it "hanoi". NOTE: the "<username>" name refers to your login name.

[ ] When you exit "msconfig" you will notice that the name of the measurement system you created shows up. Enter this measurement system by pushing the appropriate softkeys.

AVAILABLE HELP!!!

[ ] This is a very important point. Inside the measurement system (when you have entered the emulation software), you have detailed on-line help for the commands of the emulator and analyzer. This is a primary source of information during the lab and also as you use 64000-UX later. This help contains the following information sections:

Syntax
Function
Default Values
Examples
Parameters
Description

You can access this help by typing the word "help" on the command line and pressing the appropriate softkey for the help you desire. As an exercise, find the information concerning ending out of the measurement system by typing the word "help", pushing the "end" softkey and hitting the carriage return key.

Modifying the emulation configuration

[ ] The first task to do once you enter an emulator is to define its specific configuration. This can be done in a few ways. The first is to load an already defined configuration file. The second way is to......well why don't you find out! HINT: use the on-line help facilities; specifically type the command "help" and see what happens.

[ ] Modify the configuration so that it appears identical to the one shown on the next page. Save this configuration in a file called "main". Ask the instructor for help if you need it.

NOTE: when you modify the emulation configuration, the questions ask will match the ones shown below, but may not be in the exact same order.

☐ Emulation Lab

```
#
# LSD:@(#) 3.11 86/09/23
# @(mktid) 01.00
#
#
# This is an emulation configuration file

BEGIN MEMORY MAP
default guarded
0H thru 01FFFH emulation ram
02000H thru 02FFFH emulation ram
END MEMORY MAP
Micro-processor clock source ?   internal
Micro-processor clock rate greater than 10 MHz ?   no
Restrict to real-time runs?   no
Number of significant address bits?   24
Break processor on write to ROM?   yes
Interlock emulator DTACK with user DTACK ?   no
Enable Bus Error on emulation memory accesses ?   no
Enable emulator processor interrupts?   yes
Enable emulator DMA transfers?   no
Trap number for software break (0..0FH)?   0000FH
Enable emulator use of INT7?   yes
Enable Interrupt level 7 sharing ?   yes
Enable tristate delay on halts ?   no
Enable break on halt line?   enable
Enable polling for simulated I/O?   no
Simio control address 1?   SIMIO_CA_ONE
Simio control address 2?   SIMIO_CA_TWO
Simio control address 3?   SIMIO_CA_THREE
Simio control address 4?   SIMIO_CA_FOUR
Simio control address 5?   SIMIO_CA_FIVE
Simio control address 6?   SIMIO_CA_SIX
File used for standard input?   /dev/simio/keyboard
File used for standard output?   /dev/simio/display
File used for standard error?   /dev/simio/display
```

If problems arise later in the lab with the 68000 emulator, make sure your configurations match the one shown above EXACTLY!!

[ ] As a test to see if the file was stored, use the "load" command to reload the emulation configuration. This should be the exact configuration you just created.

☐ Emulation Lab

Loading the Absolute

[ ] The absolute file you created above was named "main.X". Use the "load" command to place this program into emulation memory.

[ ] As you noticed, a message came up and said "Rebuilding symbol data base". This was the edbuild command creating the symbol file (.Y) used by the emulator and analyzers in 64000-UX. This command should be run after linking the absolute for convenience, but will be run automatically when (1) it doesn't exist or (2) when the user specifies.

[ ] Use the "display registers" command and look at the program counter to determine what the transfer address of the absolute is. This will be the PC register and should have a value of 1034h. If your value differs by a small amount, it probably will create any problems. If it differs by a large amount, attempt to determine what the problem is.

FEATURES OF EMULATION

Emulation Run Control

[ ] Use the command "run from transfer_address" to begin executing the code. You could also substitute the specific address for the transfer address if you were debugging code. Look at the other features available under the run command; specifically the "until" construct of the run command. Try this out and see what happens. You have the flexibility to use an "address", "data", or "status" break point in the case of the "until" option.

[ ] Press the "break" key. Now use the display registers command like you did before. You will now see one of the nicer features of the emulator, namely program stepping. Use the step command to begin stepping through the code beginning from the transfer address. Notice how the registers change as a step is performed. Try out the different options available to the step command. For more information refer to the on-line help utility.

[ ] Differences between the "break" and "reset" keys
reset - actually pulls the reset line of the processor and puts it into a reset condition.
break - issues an NMI to the processor and saves the registers for further use. You have the ability to resume running the user program when this option is used.
For more information refer to the emulation manual.

☐ Emulation Lab

[ ] The "trace" command and "stop_trc" command are available when using the 64302 internal analyzer. Perform the following measurement:

```
trace about data 34h status write
run from transfer_address
```

You will notice that the trace will finish. Use the "display" command to see the results of the trace. The power of the analyzer has not yet been fully shown.

[ ] Issue a break to the emulator and force it into the foreground monitor.

[ ] Perform the same trace again but add the option to break into the monitor when the trigger condition is found. If you didn't issue the "break" command before setting up the new measurement, the trace would have completed automatically and at the wrong point since the emulator continued to run the code after the last trace. The analyzer merely "bus steals" the needed information and does not interfere with the operation of the emulator unless the user desires.

After the trace has completed, the execution of the user code will be temporarily stopped. Use the "display" command to look at the memory locations starting at 2000h in a blocked words format. You will notice that you stopped the Towers of Hanoi program as it was playing the game of moving the discs from one set of memory locations (or pegs) to another. By breaking into the monitor instead of using the reset feature, you have also saved the program counter.

[ ] Enter the following commands and see what happens!

```
display memory 2000h repetitively blocked words
run
```

You will notice that the game has started by placing all the discs on the top "peg". You happened to have used the trace command as to catch the user program initializing the game. You should also notice that using the "break_on trigger" specification is very helpful to "freeze" your program. What a debugging feature!

☐ Emulation Lab

Display / Modify features

[ ] You have already been introduced to the display registers, display memory and display trace commands, but you haven't yet noticed the different formats which are available. Take some time to read the help message on "display" and then play with displaying data in different formats. (10 minutes).

[ ] You also have the ability to modify both the registers and memory locations. Normally the only problems you have when modifying memory are associated with modifying ROM locations and guarded memory. Modify the emulation configuration and ADD a section of memory which is ROM. Try to modify and display these locations and see what happens. You should be able to modify these locations if they are mapped to emulation, but never if they were mapped to target.

[ ] The next display feature would be displaying symbols. The first one you should try is global symbols. Determine how to do this using the help facilities.

[ ] Another display option as far as symbols are concerned is being able to display local symbols. Use the help facilities to display the local symbols in the module "pause.P". You will have to use the syntax "display local_symbols_in <source filename>:".

[ ] Take notice of the symbol "DELAY". You can use this information for symbolic debugging. For instance, setup a trace specification to trigger about the detection of the DELAY symbol and have the analyzer break into the monitor. Display the trace data with the option for source code turned on in inverse video. You will see that the source line referencing feature of 64000-UX can be very helpful, because you can see the high-level source shown with the disassembled absolute code together in one listing.

☐ Emulation Lab

Tracing

[ ] The biggest feature of emulation is that it is coupled with the internal analyzer (emulation bus analyzer) as one module. In this role, the analyzer can be used as a major debugging tool for the user. The feature set of the internal analyzer is quite large and extensive. BRIEFLY read the help information on the trace command to get an idea of this feature set. The flexibility of the analyzer manifests itself in the trace specifications and display options. Take about 30 minutes to get a feel for what the analyzer can do, especially the "trace only" option.

[ ] You will notice that three different data types can be traced upon. These include address, data, and status. In conjunction with these features, you can specify "don't care conditions" for the trace specification. Try out this feature of the analyzer. The "don't care" character that should be used is an "x". Ask the instructor for help if this is not clear.

[ ] One neat example of the flexibility of the analyzer is to trace looping routines used as timing loops in software. As an example, display the local symbols of the pause.P module again and take note of the symbols. If you were to trace the OUTER_COUNT label and then display the trace using a count relative format, you can get an idea of how long this pause routine takes. HINT: the trick to doing this measurement is to use the "only" specification in the trace. You should get an answer of about 4.006 mS for the outer loop to execute. This would mean that the pause routine would take about 500 times that long to complete, because of the DELAY variable.

Exiting Emulation

[ ] Exit the emulation session by "end releasing" the system. If you did an msstat, you would see that the emulator is now available for use by other students.

HP 64000-UX Emulation Lab (Part 2)

Objectives:

  * debugging with emulation and the internal analyzer

Prepare absolute code

[ ] Make sure you are in pmon.

[ ] A command file for compiling all of the necessary files has been provided for you. Take a look at comp_hanoi_1.

[ ] Edit this file and add the two lines shown below. If command files are used, they will normally contain the linker command as well as the assembler / compiler commands. TAKE NOTE: the "edbuild" command is included here so that it doesn't have to be generated when entering emulation. This is usually a good idea so that you are not forced to wait for your code to compile and link and then wait for the symbol database to be built at a different time.

```
lnk -ox all_1.K > main.map
edbuild main
```

[ ] Execute the comp_hanoi_1 command file from within pmon and verify that it executed without any problems.

☐ Emulation Lab

Emulation and Debugging

[ ] Enter the emulator loading the emulation configuration used earlier as well as the new absolute (main.X). Repetitively display memory starting at 2000h and run the user code from the transfer address. Be sure to use the "blocked words" option to the display command.

Did you notice any problems?
These are the problems that you must fix. First, you noticed that the data structure for one of the discs is incorrect. Secondly, you may have noticed that the program does not appear to be running correctly.

HINTS: if you were the author of this code you would that the data structure for this programs was loaded around 021A0h. You should also notice that all the appropriate discs are showing up on the display memory, it just appears that they are being put in the wrong locations. Upon a closer examination you may notice that the program seems to be functioning properly as far as where it stacks the discs and how it stacks them.

Fix the code such that you can see the program play the game properly with a "display memory 2000h repetitively blocked words".

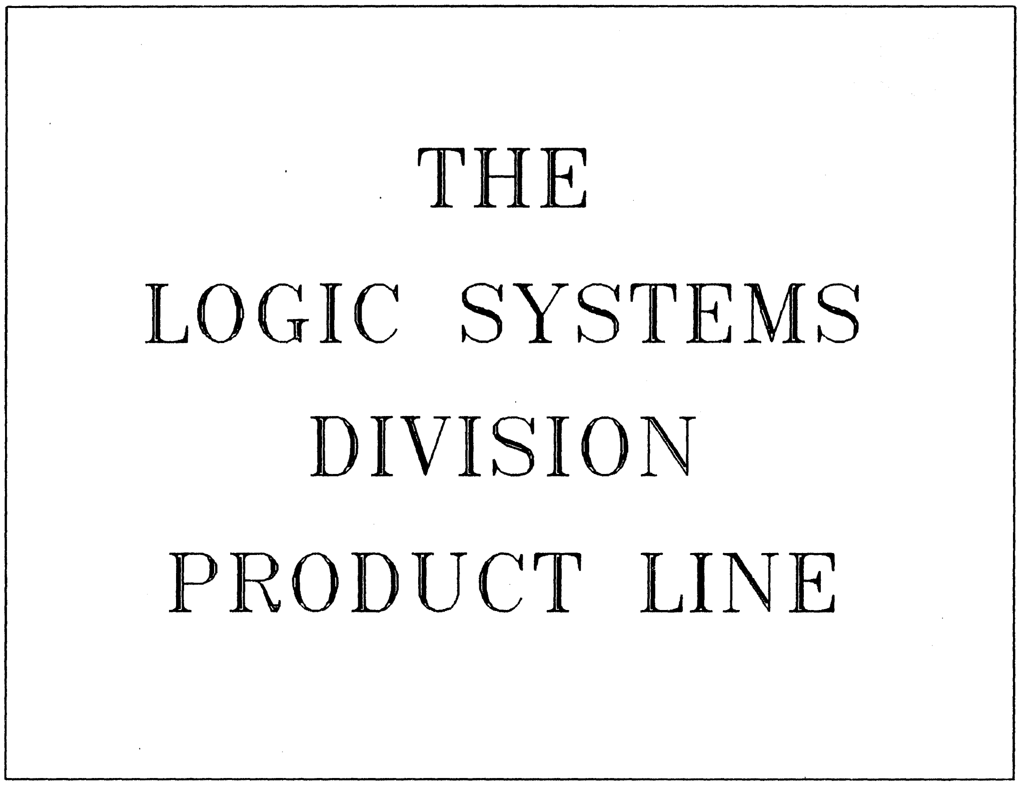
Good Luck!!

Extra Credit:

If you are feeling confident about your skills with the emulator and analyzer, try running the command file called: comp_hanoi_2. Edit this file and add the linker command and the edbuild command.

If you don't find that much of a challenge, the run the command file called: comp_hanoi_3. Don't forget to add the linker command and edbuild command to the command file.

□ LSD Product Line



THE

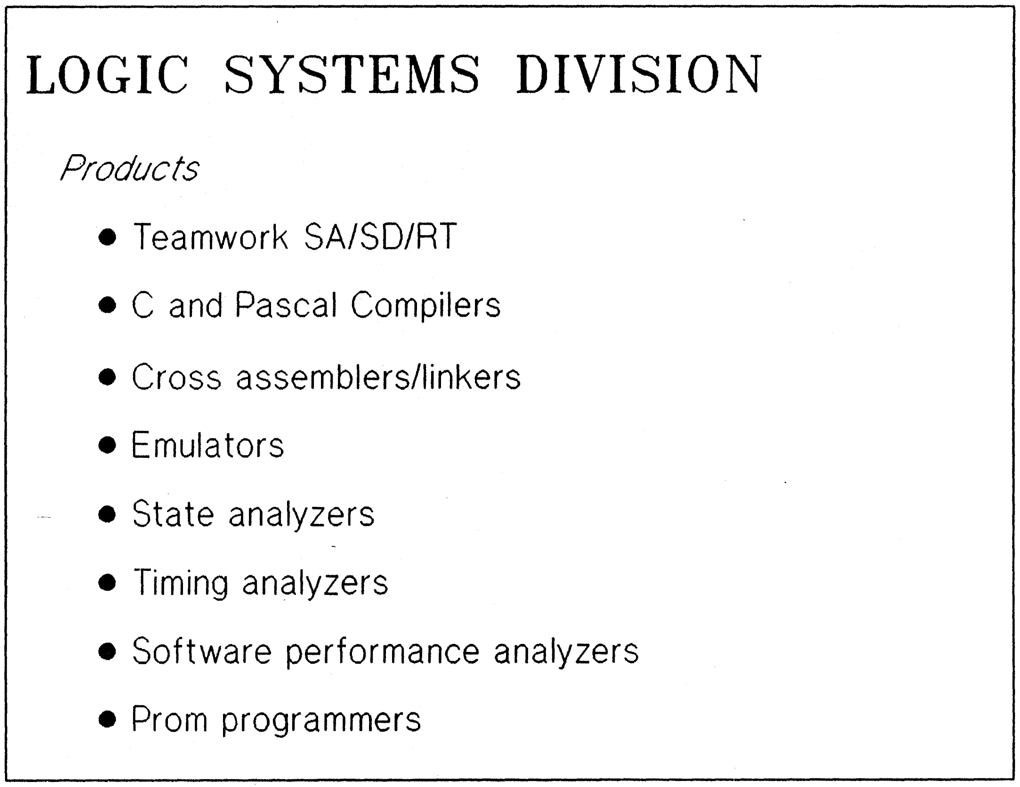
LOGIC SYSTEMS

DIVISION

PRODUCT LINE

Sll 1.35

☐ LSD Product Line



LOGIC SYSTEMS DIVISION

Products

- Teamwork SA/SD/RT
- C and Pascal Compilers
- Cross assemblers/linkers
- Emulators
- State analyzers
- Timing analyzers
- Software performance analyzers
- Prom programmers

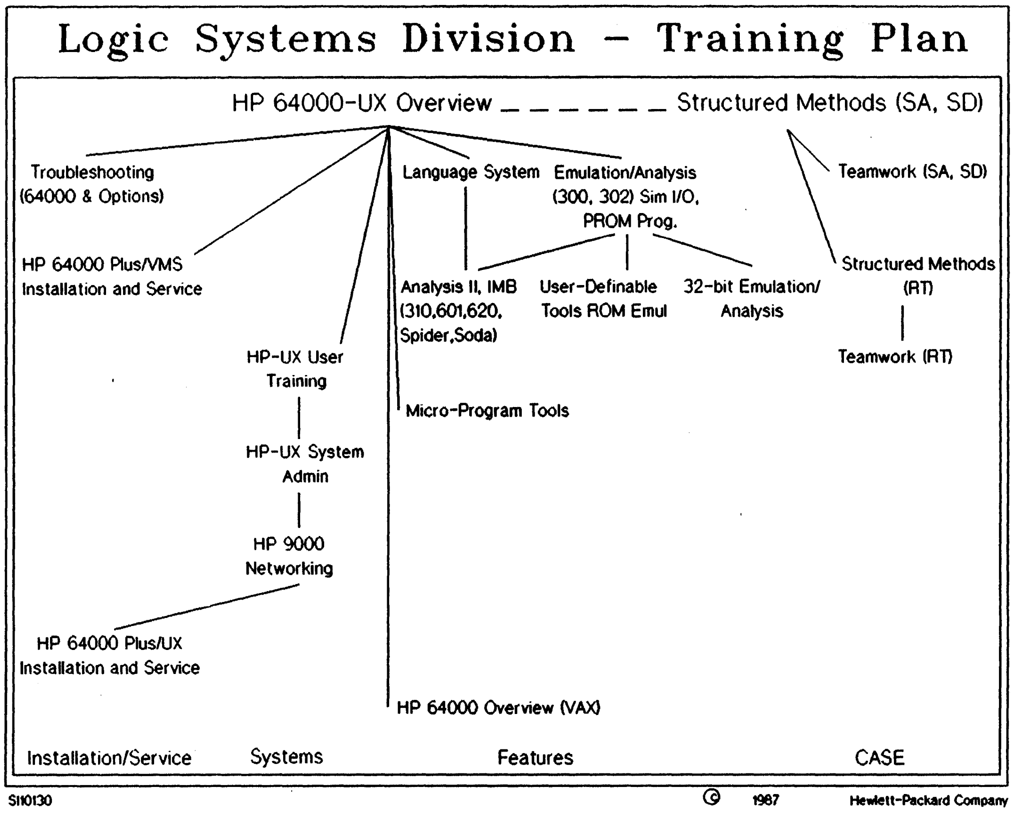
Sli 1.25 © 1987 Hewlett-Packard Company

NOTES:

Your HP sales representative (FE) can give you a complete breakdown of all the products produced and sold by Logic Systems Division.

STUDENT NOTES:

LSD Product Line

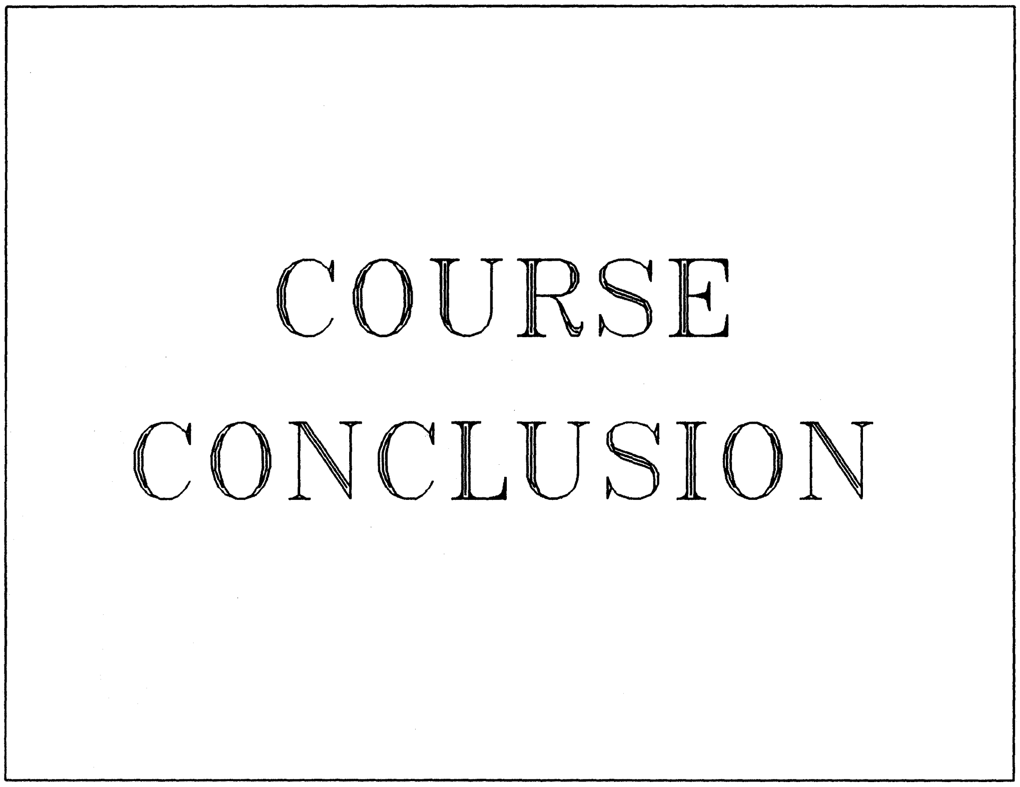

Logic Systems Division – Training Plan
HP 64000-UX Overview
Structured Methods (SA, SD)
Troubleshooting (64000 & Options)
HP 64000 Plus/VMS Installation and Service
HP-UX User Training
HP-UX System Admin
HP 9000 Networking
HP 64000 Plus/UX Installation and Service
Language System
Emulation/Analysis (300, 302) Sim I/O, PROM Prog.
Analysis II, IMB (310,601,620, Spider,Soda)
User-Definable Tools ROM Emul
32-bit Emulation/ Analysis
Micro-Program Tools
HP 64000 Overview (VAX)
Teamwork (SA, SD)
Structured Methods (RT)
Teamwork (RT)
Installation/Service
Systems
Features
CASE
S110130
© 1987 Hewlett-Packard Company

NOTES:

This slide diagram shows the training classes which are currently available or will be available for customers. As you will see, this overview course is only the first of many courses you may elect to take in a variety of subject areas.

STUDENT NOTES:

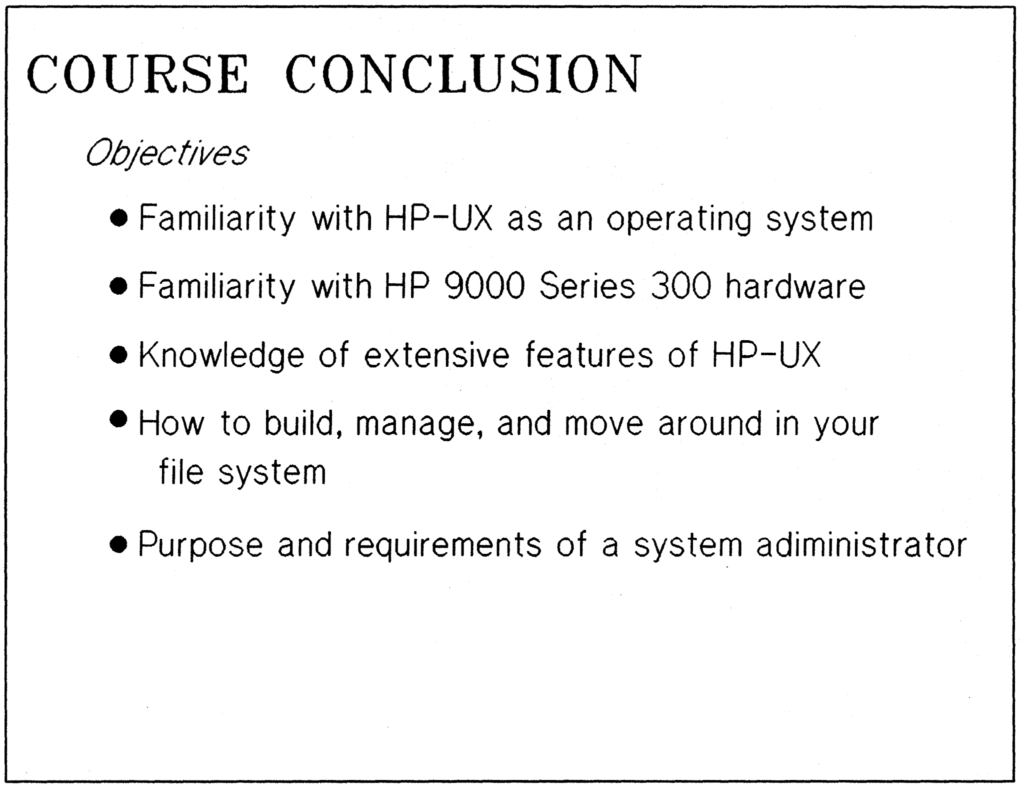
☐ Course Conclusion



COURSE
CONCLUSION

CC1 1.05

☐ Course Conclusion



COURSE CONCLUSION

Objectives

- Familiarity with HP-UX as an operating system
- Familiarity with HP 9000 Series 300 hardware
- Knowledge of extensive features of HP-UX
- How to build, manage, and move around in your file system
- Purpose and requirements of a system adiministrator

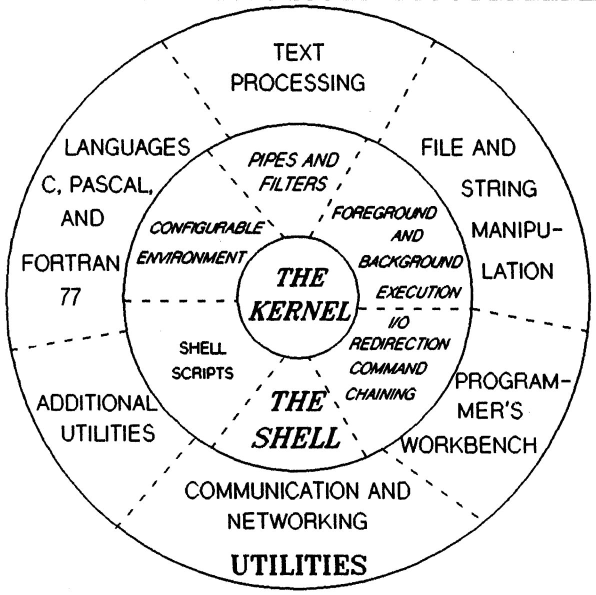
CC1 1.10

STUDENT NOTES:

☐ Course Conclusion

COURSE CONCLUSION

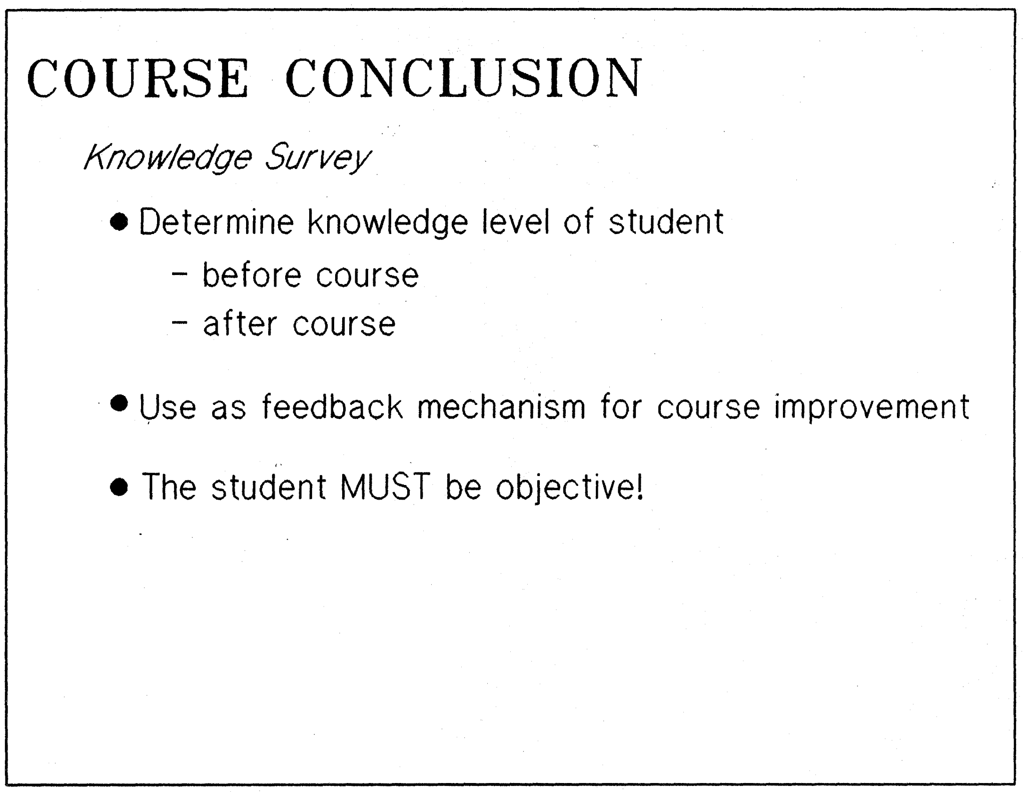
What This Course Introduced:



CC1 1.15 © 1987 Hewlett-Packard Company

STUDENT NOTES:

☐ Course Conclusion



COURSE CONCLUSION

Knowledge Survey

- Determine knowledge level of student
  - before course
  - after course
- Use as feedback mechanism for course improvement
- The student MUST be objective!

CC1 1.20

STUDENT NOTES: