

DATA SYSTEMS MARKETING
SOFTWARE TRAINING
USER TECHNIQUES

WEEK 1

	<u>MONDAY</u>	<u>TUESDAY</u>	<u>WEDNESDAY</u>	<u>THURSDAY</u>	<u>FRIDAY</u>
TOPICS	INTRODUCTION COURSE SYSTEM MACHINE	ELEMENTARY TEXT EDITOR SEGMENTER	FORTRAN INTRO SYMBOL TRACE	FORTRAN EXTENSIONS CHARACTER LOGICAL	I/O INTRINSICS

STACK VIDEO

COMMANDS
MPE
FILE



WEEK 2

TOPICS	FILE SYSTEM INTRINSICS ADVANCED TEXT EDITOR	STAR FINAL FORTRAN LAB	BASIC	BASIC	BASIC
--------	--	------------------------------	-------	-------	-------

READING ASSIGNMENTS-USER TECHNIQUES

DAY 1 - MPE, Chapter 3,4,5

DAY 2 - TEXT EDITOR, Chapter 1, 2, 3
MPE, Chapter 7

DAY 3 - FORTRAN, Chapter 1,2,3,4
SYMBOL TRACE

DAY 4 - FORTRAN, Chapter 5,6,7
FORTRAN, Chapter 8,9,10,11

DAY 5 - MPE, Chapter 6
TEXT EDITOR, Chapter 4

DAY 6 - STATISTICAL ANALYSIS ROUTINES

DAY 7 - BASIC REFERENCE
Introduction (I1 - I17)
COMMANDS (2.45-2.59)
Pocket BASIC
COMMANDS (7-8)
BUILT-IN-FUNCTIONS (7-8)
ASCII TABLE (BACK COVER)

DAY 8 - BASIC REFERENCE
Section 5, STRINGS (1-22)
Section 7, DEBUGGING (7-17)
Section 3, ARRAYS (1-21)
Pocket BASIC
DEBUGGING (38-39)

DAY 9 - BASIC REFERENCE
Section 8, FILES (1-38)
Section 10, SEGMENTATION (1-15)



HP 3000

**computer
system
introduction**

stan shell

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

SUPPORTS INTERACTIVE PROCESSING

SUPPORTS BATCH PROCESSING

SUPPORTS MULTIPLE CONCURRENT USERS

SUPPORTS CONCURRENT INTERACTIVE/BATCH

PROVIDES TIME-SHARING

PROVIDES USER ORIENTED FILE SYSTEM

HP 3000

HP 3000 CENTRAL PROCESSING UNIT

■ ROM MICROPROCESSOR — 175 ns CYCLE

■ 170 UNIQUE INSTRUCTIONS

■ ALL INSTRUCTIONS SINGLE WORD (16 BITS)



■ INSTRUCTION LOOK-AHEAD

■ INSTRUCTIONS FOR BIT, BYTE, WORD AND DOUBLE WORD MANIPULATION

■ INTEGER, DOUBLE INTEGER, AND FLOATING POINT HARDWARE ARITHMETIC

■ INSTRUCTIONS DESIGNED FOR EASY COMPILER IMPLEMENTATION

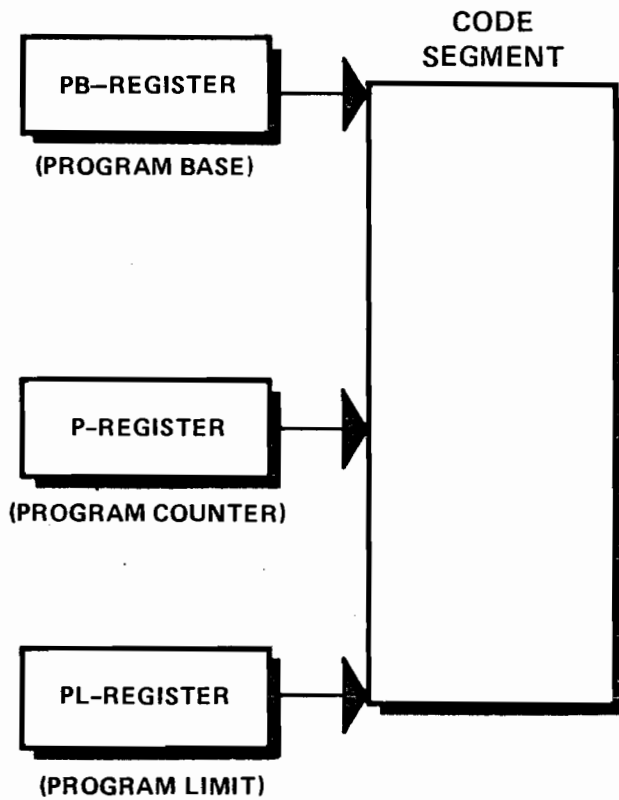
■ INSTRUCTIONS TO PERFORM MANY OPERATING SYSTEM FUNCTIONS

HARDWARE CHARACTERISTICS

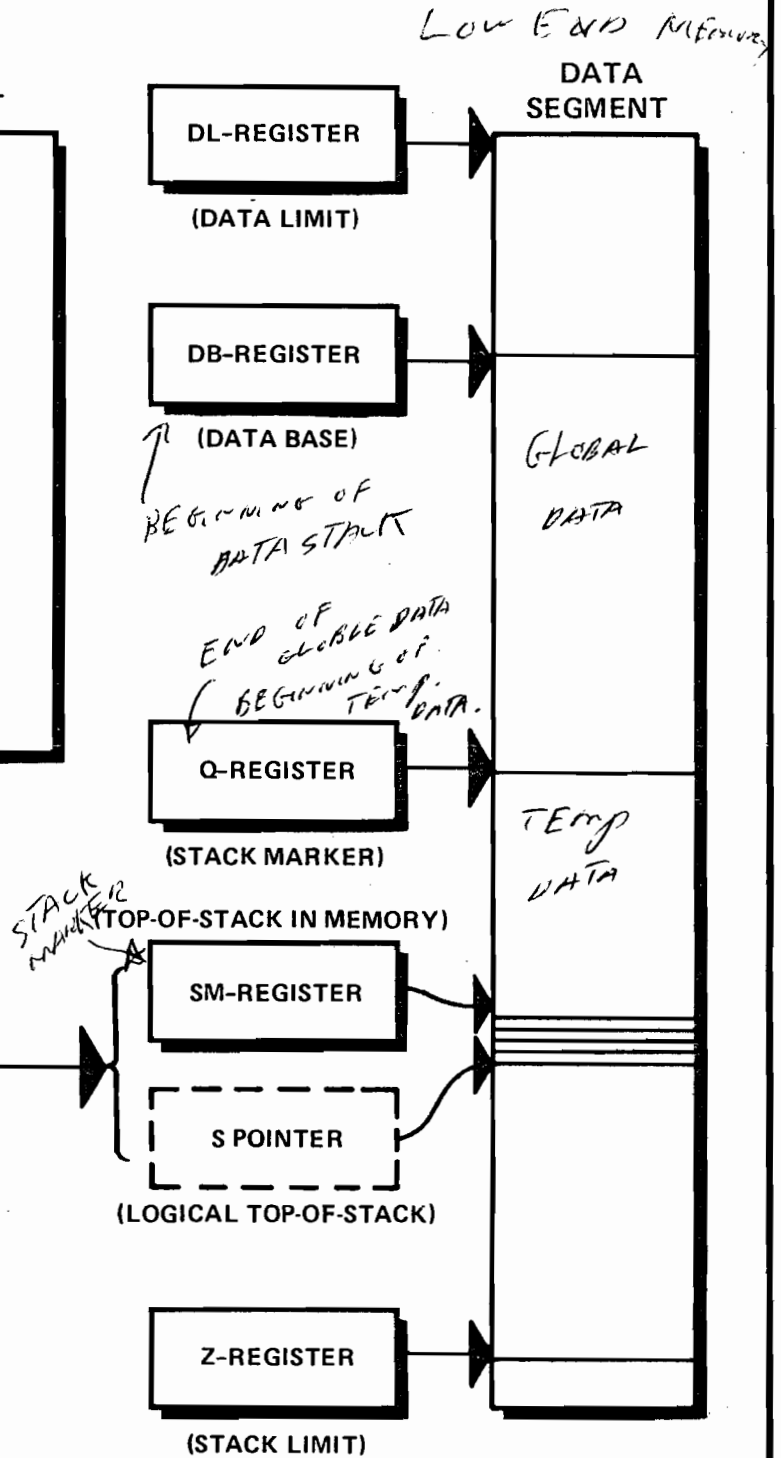
- ⌞ HARDWARE IMPLEMENTED STACK
- CODE AND DATA SEPARATION
- ⌞ VARIABLE LENGTH CODE SEGMENTATION
- REENTRANT PROGRAM CODE
- RELOCATABLE PROGRAM CODE
- ⌞ INDEX REGISTER
- VIRTUAL MEMORY STRUCTURE
- ⌞ MULTIPLE DATA FORMATS

HP 3000

CODE SEGMENT POINTING REGISTERS



DATA SEGMENT POINTING REGISTERS

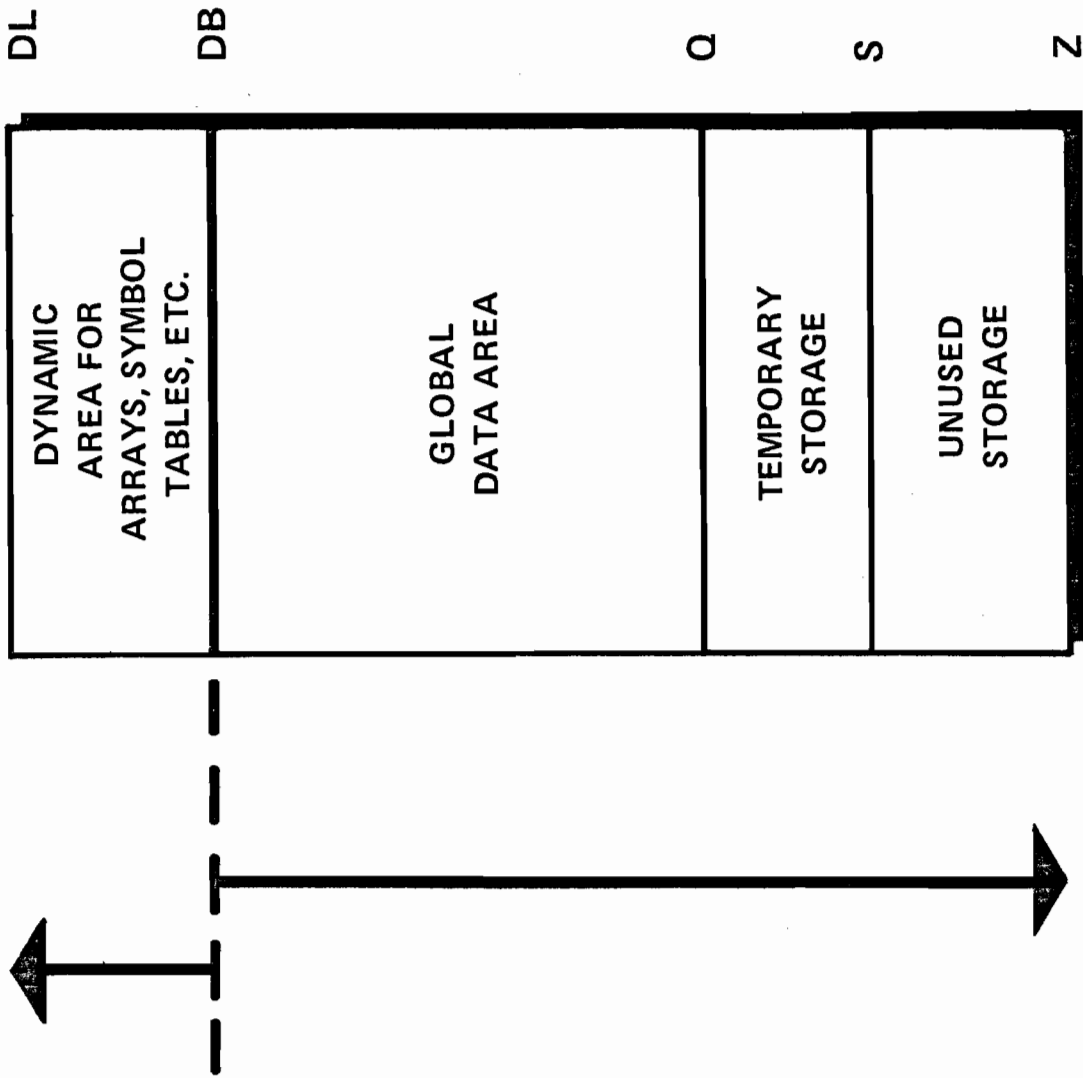


SR-REG
DISPLACEMENT
= 0, 1, 2, 3, 4

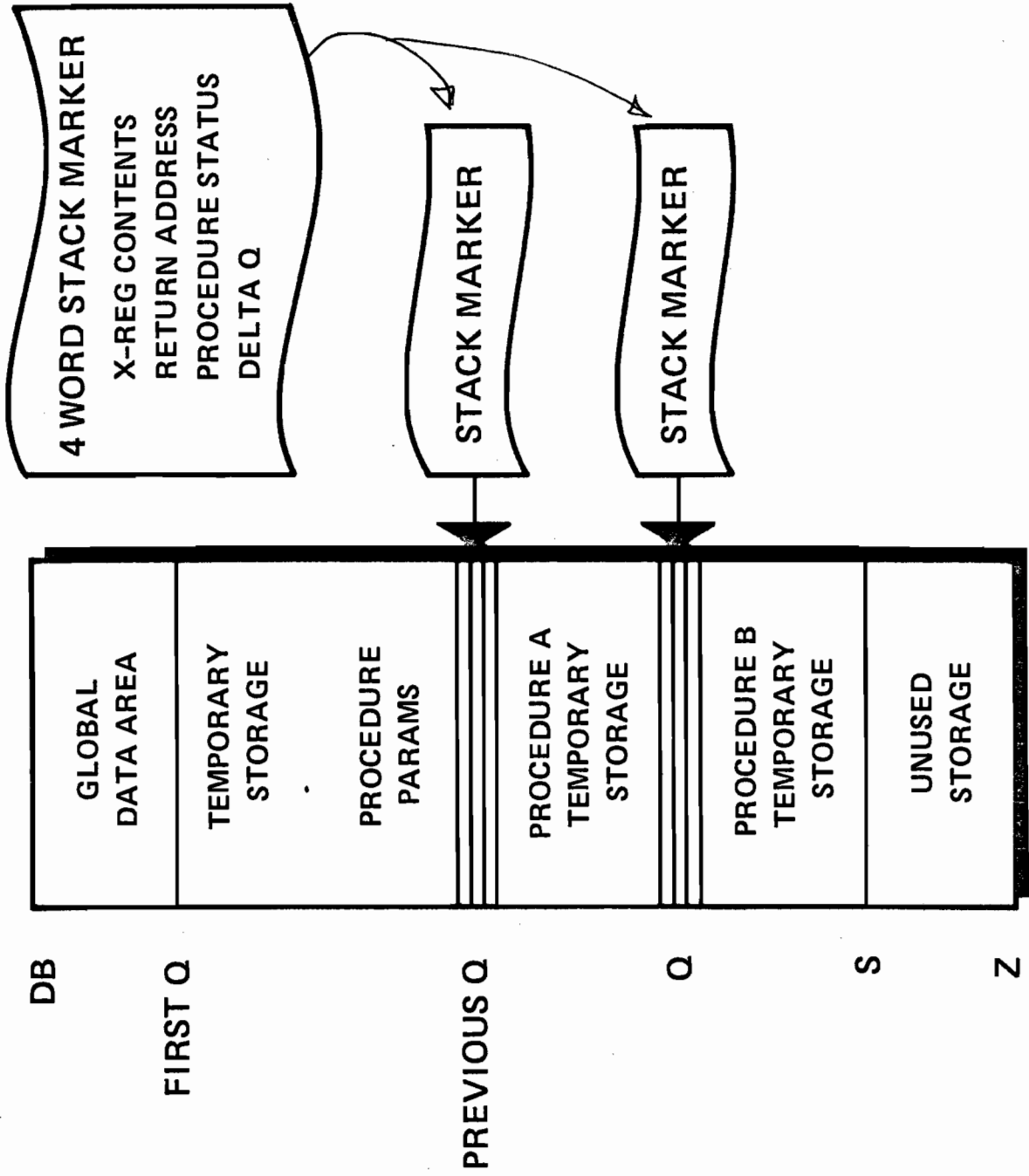
INCREASING
ADDRESSES



Stack Data Storage



Stack Marker Chain



STACK ARCHITECTURE BENEFITS



USER/SYSTEM PROTECTION

} code not written to disc.

VIRTUAL
MEMORY

ONLY READ OF DISC.



REDUCED DISC TRANSFERS



CODE COMPRESSION

ALL 16 BIT INSTRUCTIONS
BECAUSE ALL DATA STACK
REFERENCES



EFFICIENT EXPRESSION EVALUATION



POWERFUL PROCEDURE PROCESSING



DYNAMIC STORAGE ALLOCATION

HP 3000

VIRTUAL MEMORY



PRIMARY MEMORY



SECONDARY (SWAPPING) MEMORY



CODE NOT SWAPPED OUT

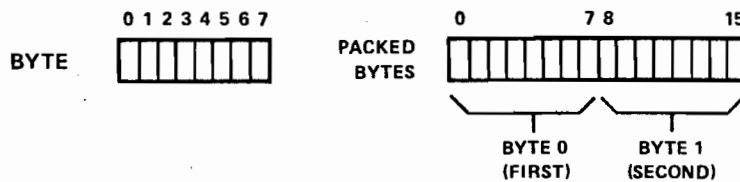


DATA IS SWAPPED

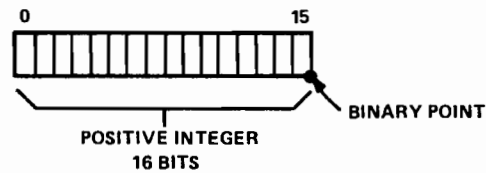


VARIABLE LENGTH CODE AND DATA SEGMENTS

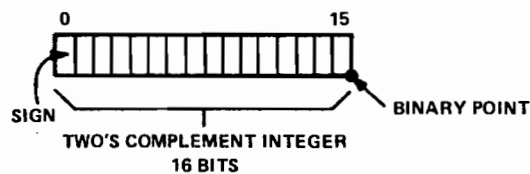
*(no PAGE-INO - MUST BE ON
PAGE BOUNDARIES)*



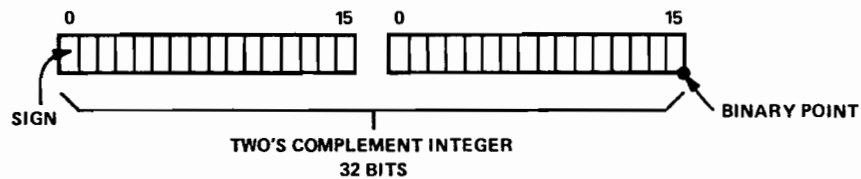
LOGICAL FORMAT



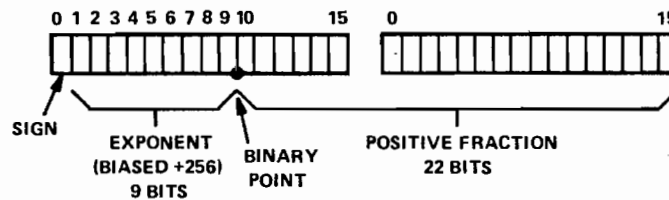
FIXED POINT
SINGLE WORD



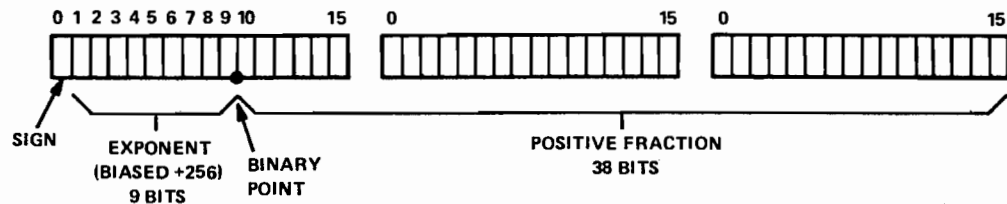
FIXED POINT
DOUBLE WORD



FLOATING POINT



FLOATING POINT
LONG FORMAT



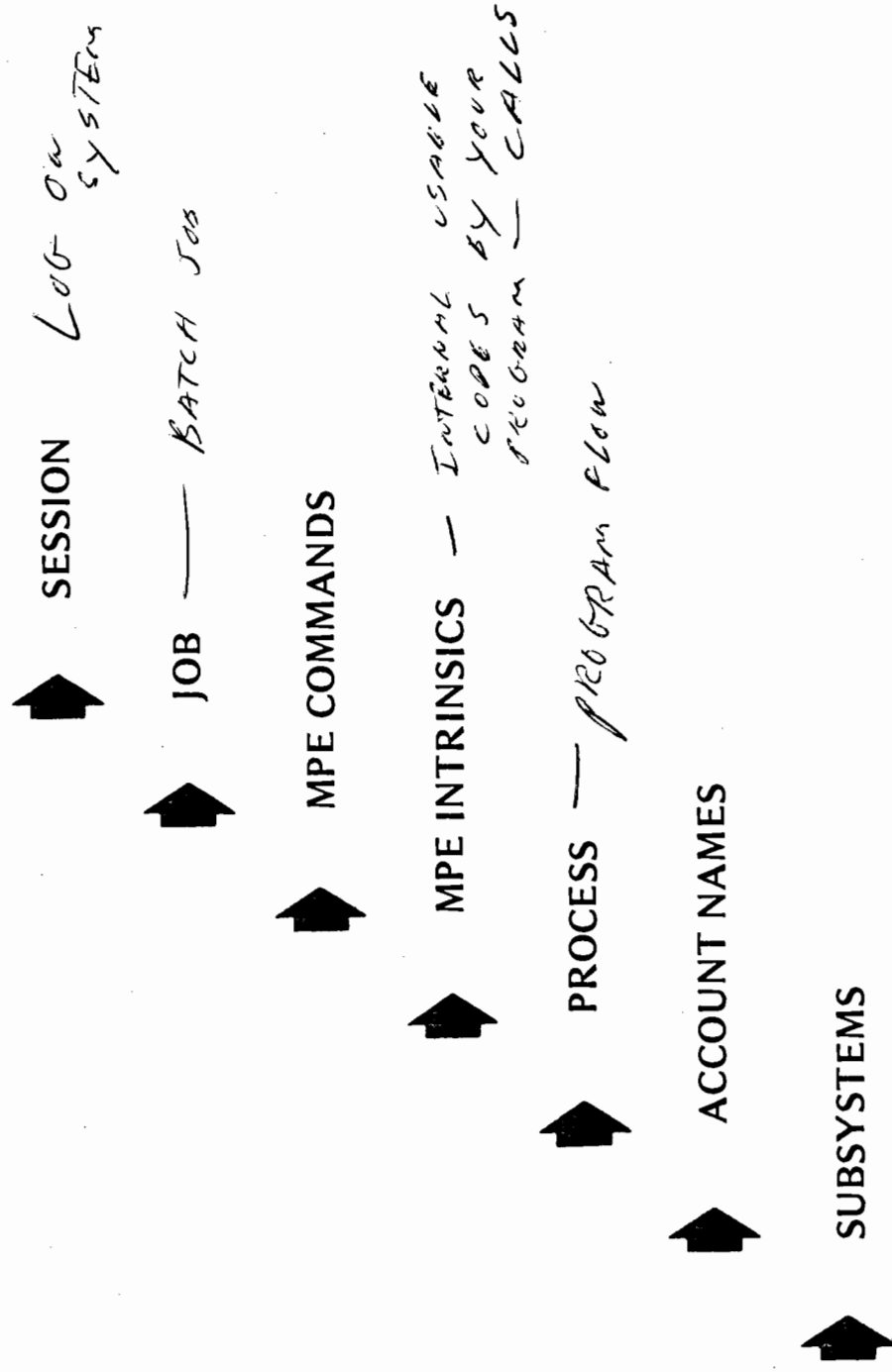
INCREASING ADDRESSES IN MEMORY



HP 3000 DATA FORMATS

HP 3000

SOFTWARE CONCEPTS



SESSION MODE OPERATION

INPUT THROUGH INTERACTIVE TERMINAL ~

START → INITIATED BY RETURN KEY ~

PROPTER → SYSTEM SUPPLIES COLON ~

NO SESSION ABORT ON COMMAND ERROR ~

→ COMMANDS AND DATA CAN BE INTERMIXED ~

Stop - Resumable Interrupt.
PROGRAMS CAN BE INTERRUPTED BY (BREAK KEY) ~

DELIMITED BY "HELLO" AND "BYE" ~

BATCH MODE OPERATION

INPUT THROUGH SERIAL DEVICE — *card reader, tape, etc.*

INITIATED BY DEVICE INTERRUPT → *Reset etc.*

USER SUPPLIES COLON →

COMMAND ERROR MAY CAUSE JOB ABORT

COMMANDS AND DATA CAN BE INTERMIXED

DELIMITED BY "JOB" AND "EOJ"



HELLO COMMAND PARAMETERS

☐ SESSION NAME

S9, cHar.

☒ USER NAME (REQ)

REQUIRED
[Signature]

☒ "PERIOD" ACCOUNT NAME (REQ)

☐ "COMMA" GROUP NAME

☐ TERMINAL TYPE

☐ PRIORITY

☐ OUTPUT LISTING DEVICE

REQUIRED

S01.UT
[Signature]

:HELLO/STAN,S9.UT,GP9;PRI=ES

3000 MPE COMMANDS

PRECEDED BY COLON

COMMAND NAME

POSITIONAL PARAMETERS

SEP for page

KEYWORD PARAMETERS

SEP keywords

CONTINUATION ALLOWED

:BUILD MASTERFL; TEMP; REC = -80,,F,ASCII

:PREP INPT,OUTP;& *CONTINUATION*

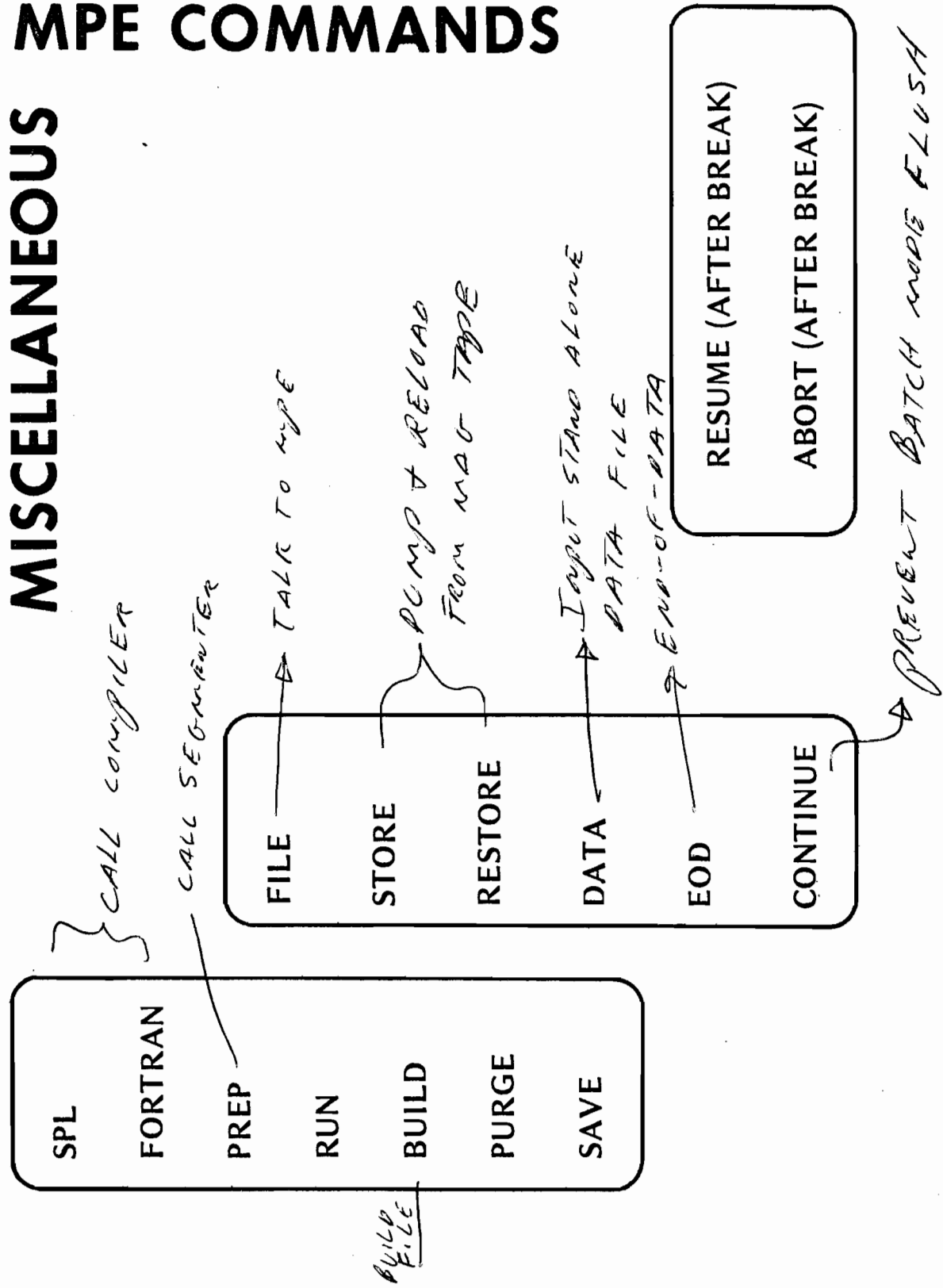
:DL = 500 ; &

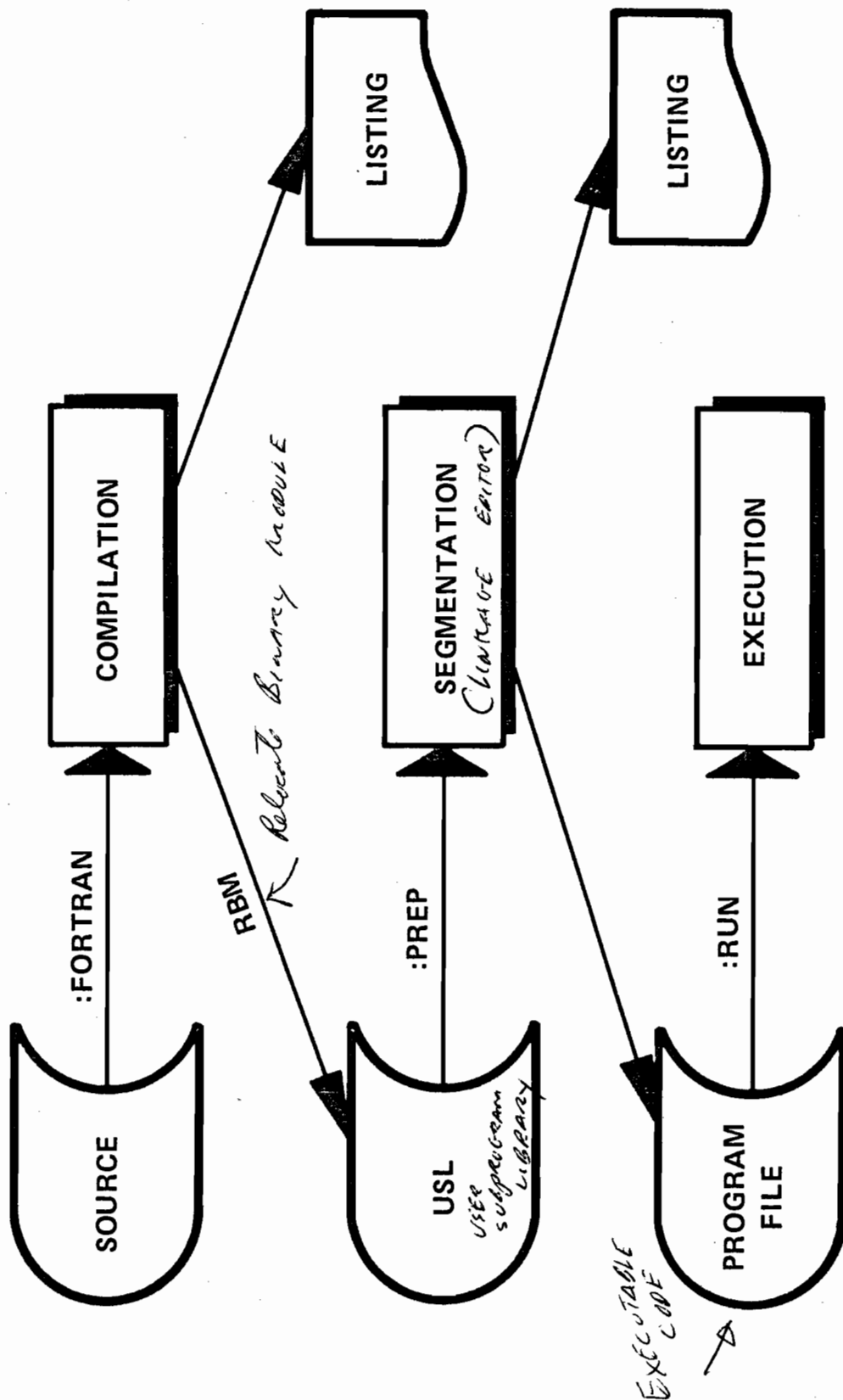
:CAP = PH,DS,MR



MPE COMMANDS

MISCELLANEOUS





MPE PROGRAM MANAGEMENT

PROGRAM EXECUTION

:SPL SOURCEFL

FILE

Input output

:PREP \$OLDPASS, \$NEWPASS

copy back

SYSTEM TEMP FILES

PREPS \$OLDPASS

CHANGES \$NEWPASS TO \$OLDPASS

:RUN \$OLDPASS

SAMPLE BATCH JOB DECK

PREVENT ON
PUSH ON
NEXT COMMAND

:RUN	:CONT	:PREP MY	:EOD	(FORT	(FORT	(FORT	\$CONTROL USL INI	:FORTAN, MYJSL
------	-------	----------	------	-------	-------	-------	-------------------	----------------

VENTION	:EDJ
ISH ON	:RUN LISTPRG
TWARD	:EOD
	(PROG DATA)
	(PROG DATA)
	(PROG DATA)
	:RUN PROG
	:CONTINUE
	:PREP MYUSL, PROG
	:EOD
	(FORTAN SOURCE DECK)
	(FORTAN SOURCE DECK)
	(FORTAN SOURCE DECK)
	\$CONTROL USLINIT
	:FORTAN, MYUSL
	:BUILO PROG; CODE = 1029
	:BUILO MYUSL; CODE = 1024
	:JOB MANAGER, SYS

[illegible][illegible][illegible][illegible]

Year	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095	2096	2097	2098	2099	2100
1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095	2096	2097	2098	2099	2100	

05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000 1001 1002 1003 1004 1005 1006 1007 1008 1009 1010 1011 1012 1013 1014 1015 1016 1017 1018 1019 1020 1021 1022 1023 1024 1025 1026 1027 1028 1029 1030 1031 1032 1033 1034 1035 1036 1037 1038 1039 1040 1041

[illegible]

τ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	
τ	0.0000	0.0001	0.0002	0.0003	0.0004	0.0005	0.0006	0.0007	0.0008	0.0009	0.0010	0.0011	0.0012	0.0013	0.0014	0.0015	0.0016	0.0017	0.0018	0.0019	0.0020	0.0021	0.0022	0.0023	0.0024	0.0025	0.0026	0.0027	0.0028	0.0029	0.0030	0.0031	0.0032	0.0033	0.0034	0.0035	0.0036	0.0037	0.0038	0.0039	0.0040	0.0041	0.0042	0.0043	0.0044	0.0045	0.0046	0.0047	0.0048	0.0049	0.0050	0.0051	0.0052	0.0053	0.0054	0.0055	0.0056	0.0057	0.0058	0.0059	0.0060	0.0061	0.0062	0.0063	0.0064	0.0065	0.0066	0.0067	0.0068	0.0069	0.0070	0.0071	0.0072	0.0073	0.0074	0.0075	0.0076	0.0077	0.0078	0.0079	0.0080	0.0081	0.0082	0.0083	0.0084	0.0085	0.0086	0.0087	0.0088	0.0089	0.0090	0.0091	0.0092	0.0093	0.0094	0.0095	0.0096	0.0097	0.0098	0.0099	0.0100

➡ A PROCESS IS THE BASIC "RUNNABLE" ENTITY MANAGED BY THE
3000 MULTI-PROGRAMMING SYSTEM

THE 3000 PROCESS

➡ A PROCESS MUST CONTAIN AT LEAST:
ONE OR MANY 1. ONE GROUP OF INSTRUCTIONS CALLED:
—A CODE SEGMENT—

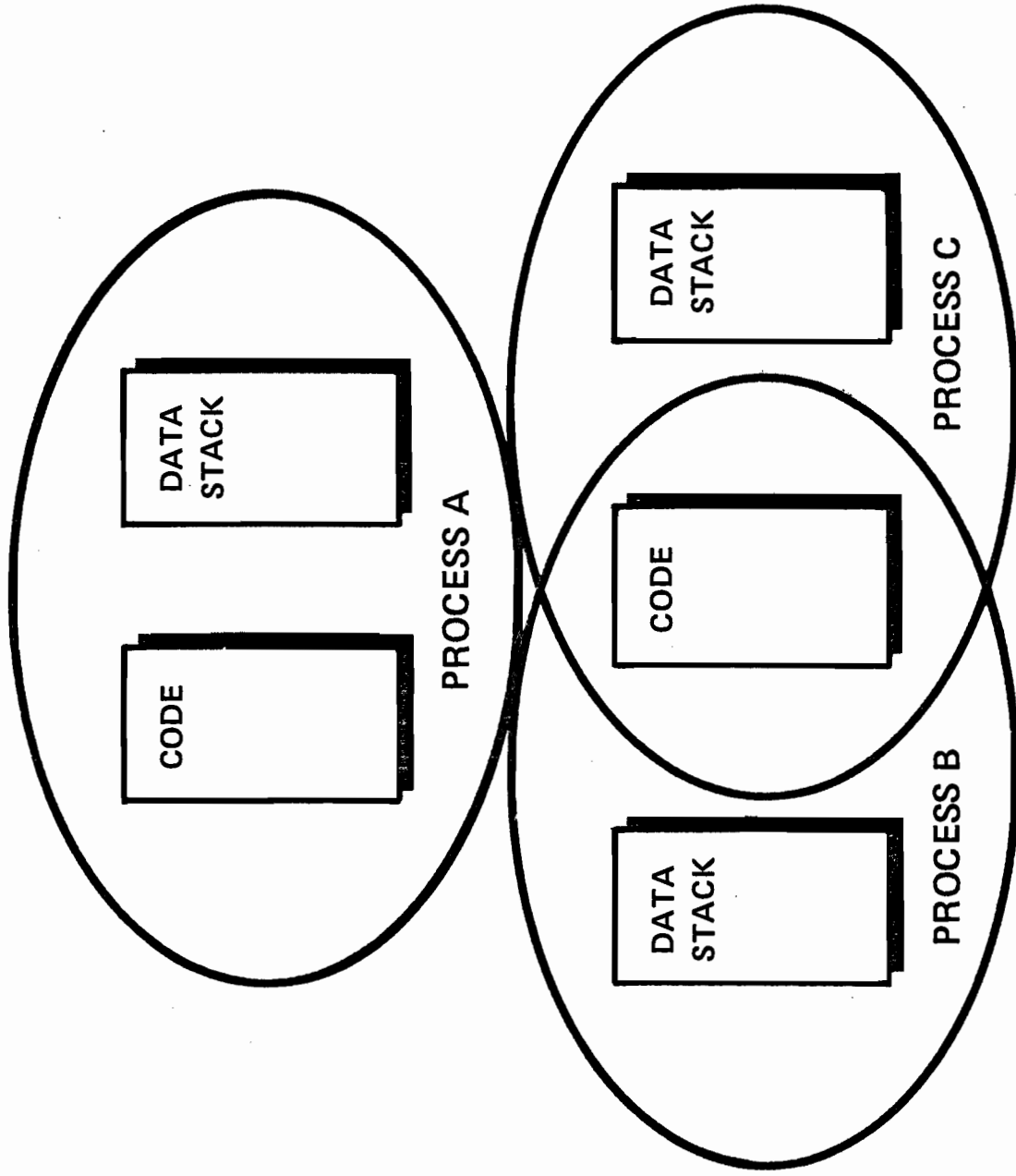
ONE ONLY 2. A DATA DOMAIN THAT IS ABSOLUTELY PRIVATE CALLED:
—A STACK—

3. A SET OF DESCRIPTIVE INFORMATION CALLED:
—THE PROCESS CONTROL BLOCK (PCB)—

➡ A PROCESS HAS A UNIQUE PROCESS CONTROL BLOCK AND STACK,
BUT MAY HAVE MORE THAN ONE CODE SEGMENT.

➡ A PROCESS IS THE FLOW OF CONTROL THROUGH PROGRAM CODE

Code Sharing by Processes



- MAJOR BILLABLE ENTITY FOR RESOURCES USED

THE ACCOUNT

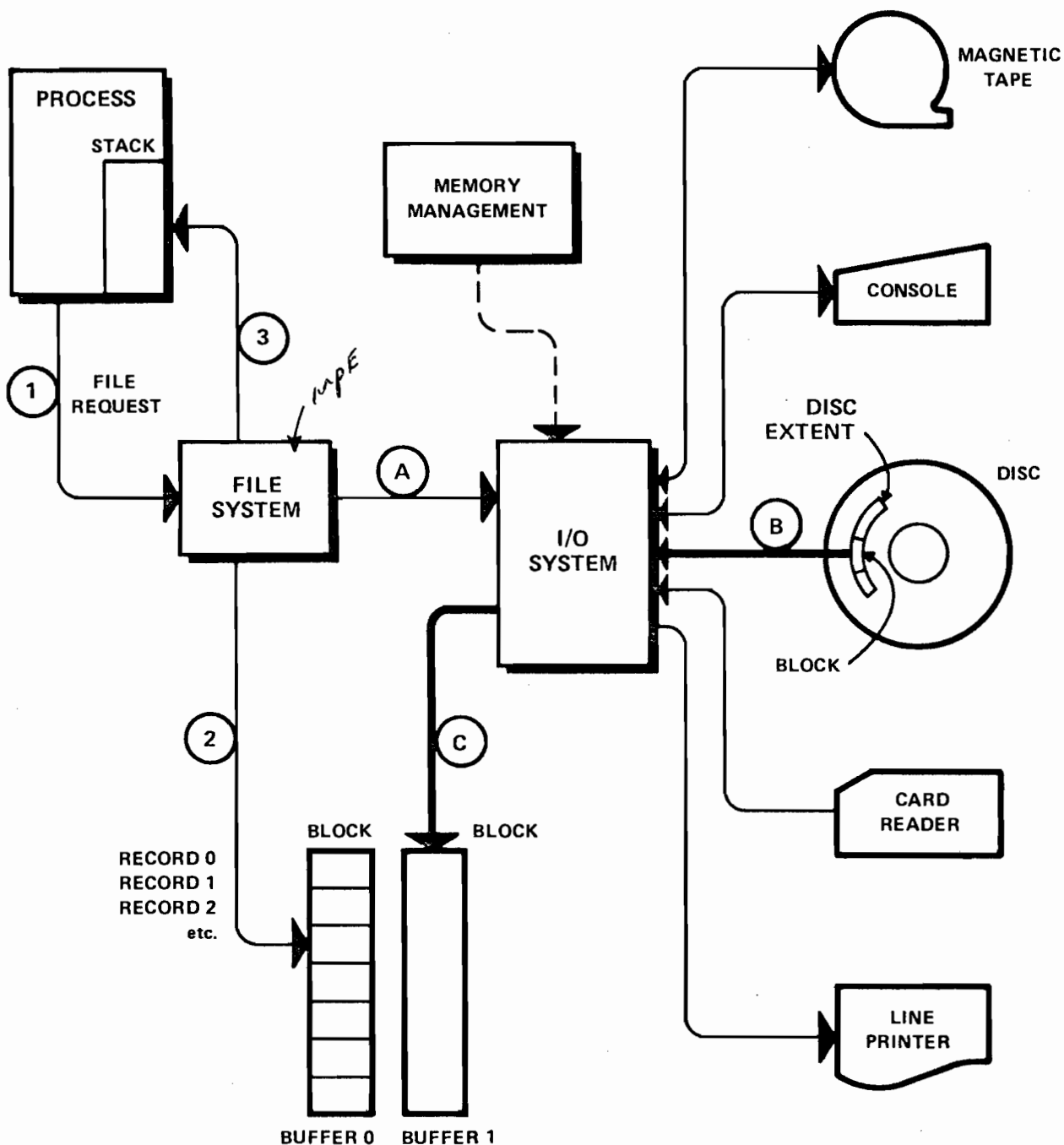
- SYSTEM MANAGER DEFINES EACH ACCOUNT

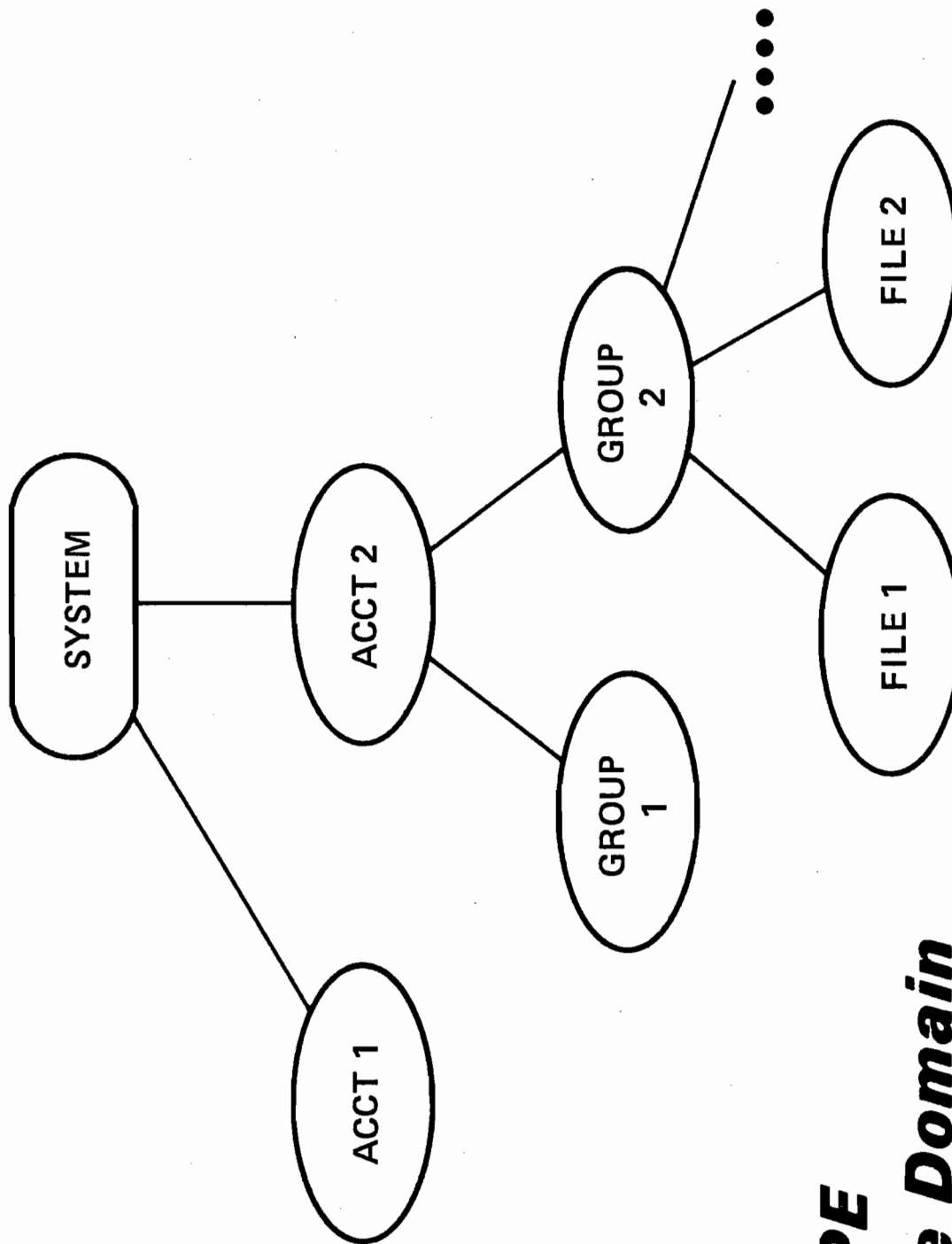
- ☐ NAME AND (PASSWORD)
- ☐ ACCOUNT MANAGER
- ☐ CAPABILITY LIST
- ☐ SCHEDULING PRIORITY
- ☐ MAXIMUM FILE SPACE, CPU TIME, CONNECT TIME
- ☐ FILE ACCESS MODES

- SYSTEM KEEPS RUNNING TOTAL OF RESOURCE USAGE FOR EACH ACCOUNT

HP 3000 FILE SYSTEM

128 words/sector





***MPE
File Domain***

FILE ADDRESSING

LOCAL



F/C₁

OUTSIDE GROUP



F/C₁.G/C₂

OUTSIDE ACCOUNT



F/C₁.G/C₂.A/C₃

where

examples

f — LOCAL FILE NAME

PAYROLL/XZ2A

g — GROUP NAME

PAYROLL/XZ2A. GRP7

a — ACCOUNT NAME

c — PASSWORD

PAYROLL/XZ2A.GRP7.ACCT47/X2AD9

HP 3000 SUBSYSTEMS

✓ SPL COMPILER

✓ FORTRAN COMPILER

✓ COBOL COMPILER

✓ BASIC INTERPRETER

✓ TEXT EDITOR

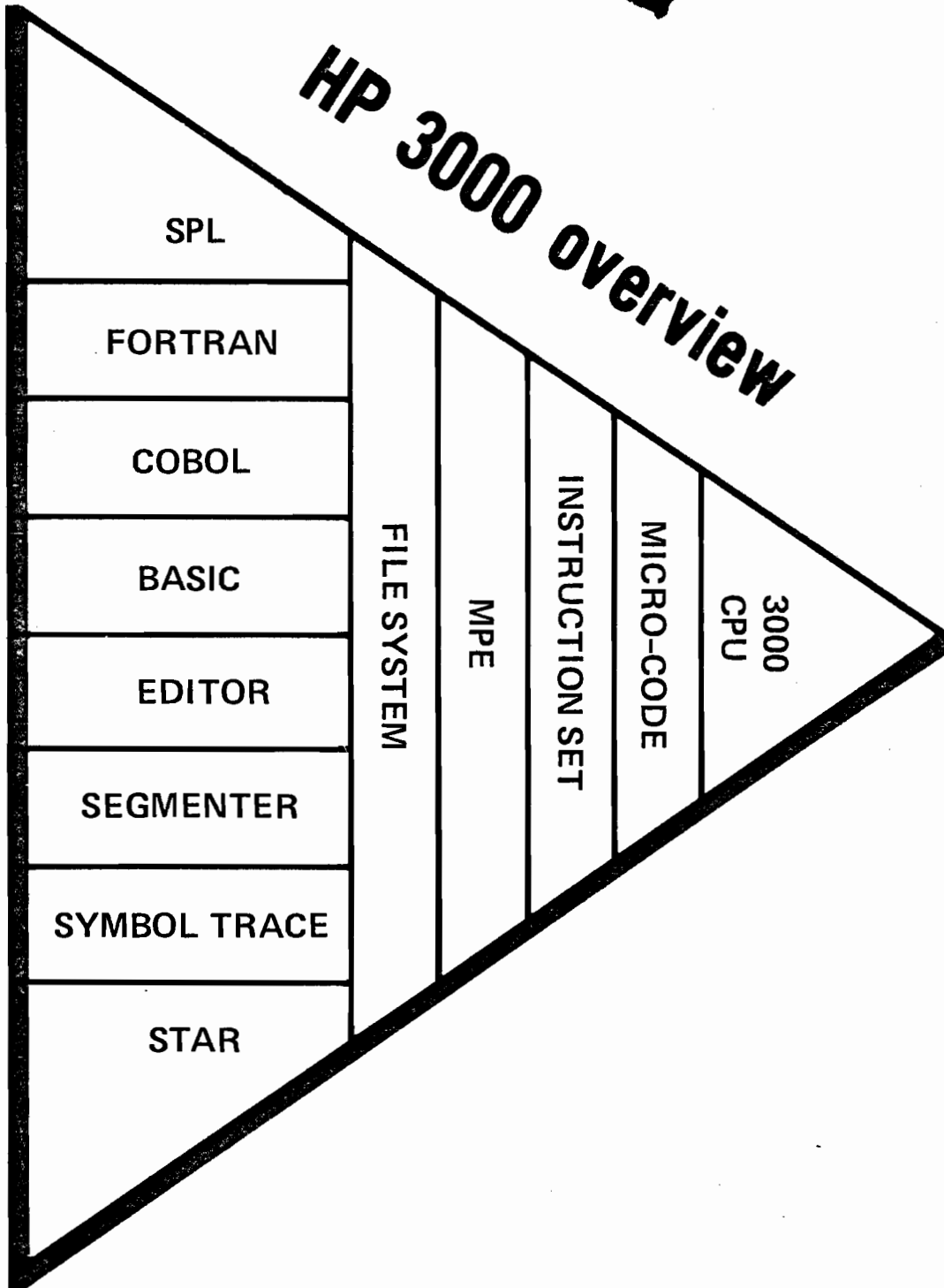
✓ SEGMENTER

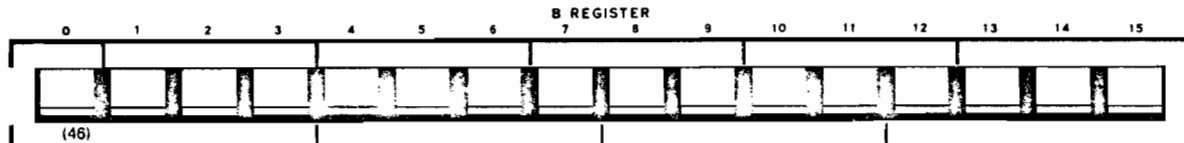
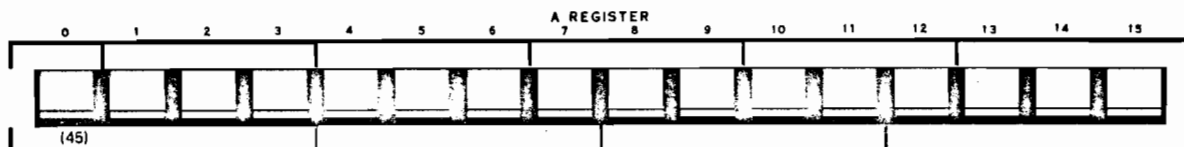
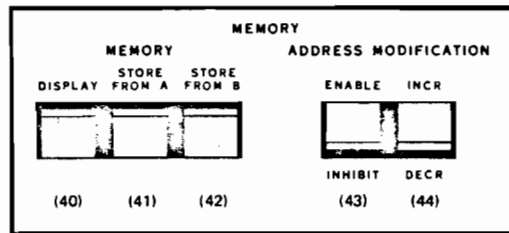
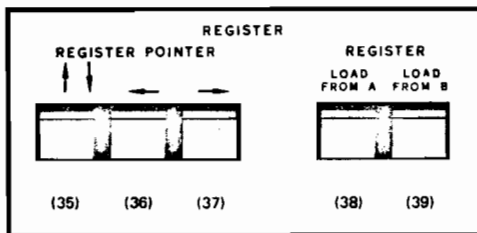
✓ STAR

✓ SYMBOL TRACE

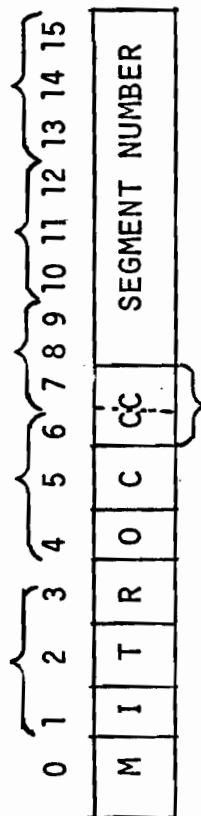


HP 3000 overview





displayed in
selects register



1 Ext Interrupts 1=Enabled

1 Ext Interrupts 1=Enabled

T User Traps 1=Enabled

R Left/Right stack op

0 Overflow

C Carry

CC Condition Code

CCL=1 LESS

CCF=2 Equal

CCG=Ø

CONDITION CODES

CCA	CCL	Operand<Ø
	CCE	Operand=Ø
	CCG	Operand>Ø

CCB	CCL	Special ASCII Char
	CCE	Alphabetic
	CCG	Numeric

CCC	CCL	Operand 1<OPR 2
	CCE	Operand 1=OPR 2
	CCG	Operand 1>OPR 2

LOAD TYPES

1. COOLSTART
2. COLDSTART
3. RELOAD
4. UPDATE

INFORMATION NEEDED

1. MPE
2. INSTALLATION (I/O DEVICES CONFIGURATION AND
ACCOUNTING INFORMATION)
3. USER FILES (INCLUDING COMPILERS)

TYPE

SOURCE

REGISTER B OCTAL

↓
REQUEST
UP

reconfiguration

COOLSTART

DISK
MPE
INSTALLATION
USER

TAPE

5
TAPPE

COLDSTART

USER

MPE
INSTALLATION

3006
PISC

RELOAD

NO USEFUL FILES
PRESENT

MPE
INSTALLATION

3006

UPDATE

USER
INSTALLATION

MPE

3006

DAY TYPE UP

SOFTWARE
UPDATE

COOLSTART

1. SET REGISTER "B" TO OCTAL 5

2. HALT

3. I/O RESET

CPU RESET

COLD LOAD — *System Temp run, THEN HALT*

4. RUN

5. LOAD MAP? (cr)

6. ANY CHANGES? (cr)

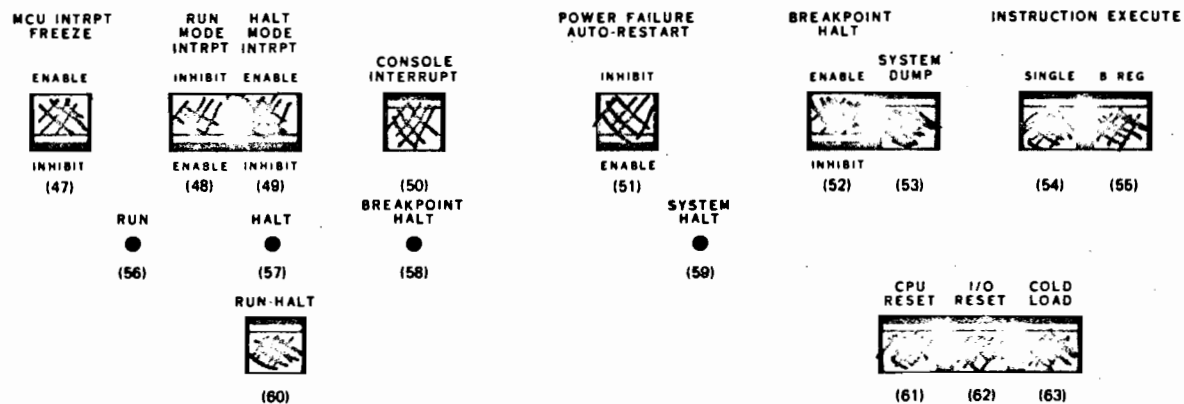
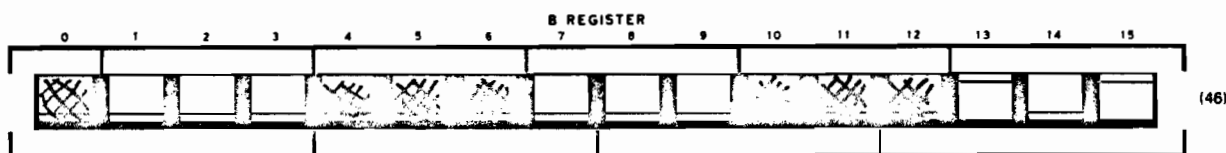
7. DATE?

mm/dd/yy (cr) *Zero fill not need.*

8. TIME?

hh:mm (cr)

(24 hour clock)



HP32000A.01.02
LOAD MAP?
ANY CHANGES?
DATE?
3/30/73
TIME?
9:40
WELCOME

INFORMATION NEEDED

1. MPE
2. INSTALLATION (I/O DEVICES CONFIGURATION AND ACCOUNTING INFORMATION)
- ; 3. USER FILES (INCLUDING COMPILERS)

LOAD TYPES

1. COOLSTART
2. COLDSTART
3. RELOAD
4. UPDATE

TYPE	SOURCE		REGISTER B OCTAL
	DISK	TAPE	
COOLSTART	MPE INSTALLATION USER		5
COLDSTART	USER	MPE INSTALLATION	3006
RELOAD		MPE INSTALLATION	3006
UPDATE	USER INSTALLATION	MPE	3006

COLDSTART

1. SET REGISTER "B" TO OCTAL 3006
2. MOUNT MPE TAPE
3. HALT
4. I/O RESET
CPU RESET
COLD LOAD — *Temp Run*
5. RUN
6. WHICH OPTIONS (COLDSTART/RELOAD/UPDATE)? *COL*
7. ENTER COLDSTART (CR)
8. LOAD MAP? (CR)
9. ANY CHANGES (CR)
10. DATE? *LARGE WAITING TIME*
MM/DD/YY (CR)
11. TIME?
HH:MM (CR) (24 HOUR CLOCK)
12. *WELCOME*

HP32000A.01.02
WHICH OPTION <COLDSTART/RELOAD/UPDATE>? COL
LOAD MAP?
ANY CHANGES?
DATE?
3/30/73
TIME?
9:44
WELCOME

RELOAD

SAME AS COLDSTART EXCEPT STEP 7 RESPONSE IS

RELOAD (CR)

REL

SUBSYSTEMS AND USER FILES NEED RESTORING

UPDATE

SAME AS COLDSTART EXCEPT STEP 7 RESPONSE IS

UPDATE (CR)

UPD



STORE

USED TO COPY DISK TO TAPE

: STORE FILESETLIST ; DESTINATION

↓
LIST OF FILES

↓
*TAPE FILE

:JOB username.account
JOB NUMBER = #J5
MON, APR 2, 1973, 8:30 AM
HP3200A.01.02
:FILE T;DEV=TAPE
:STORE &
: BASIC , &
: FORTRAN , &
: SPL , &
: STAR , &
: TAPECOPY &
: ; #1

FILES STORED = 5

FILE	.GROUP	.ACCOUNT	LDN	ADDRESS
BASIC	.PUB	.SYS	2	%000000000052
FORTTRAN	.PUB	.SYS	2	%000000004747
SPL	.PUB	.SYS	2	%000000012224
STAR	.PUB	.SYS	2	%000000012725
TAPECOPY	.PUB	.SYS	2	%000000013261

FILES NOT STORED = 0

:EOJ
MON, APR 2, 1973, 8:31 AM

Time Job process
Time Job process
I.O. II
 ? IO/17:52/#J3/22/LDEV# FOR MAST ON TAPE (NUM)
 =REPLY 22.7
in unit device
process
I.O. II

10 = MAG TAPE
TO LOOK LIKE
CARD DECK

?IO TYPE OF REQUEST

/

17:52 TIME

/

#J3 JOB OR SESSION #

/

22 PROCESS IDENTIFICATION #

/

LDEV# FOR MAST ON TAPE(NUM)

CONSOLE INTERRUPT SWITCH

=

SYSTEM PROMPT CHARACTER

REPLY NEEDED

22 PROCESS IDENTIFICATION #

,

7 LOGICAL DEVICE NUMBER

RESTORE

USED TO COPY TAPE TO DISK

IF COPY OF PROGRAM
ALREADY THERE
DO NOT OVERWRITE

:RESTORE STOREDSET[;[FILESETLIST][;[DEVICE][;KEEP]]]

↓
LIST OF FILES

↓
KEEP EXISTING DISK FILES

↓
DISK DEVICE CLASS NAME

OR

LOGICAL DEVICE NUMBER

```
:JOB username:account
JOB NUMBER = #J6
MON, APR 2, 1973, 8:31 AM
HP32000A.01.02
:FILE T;DEV=TAPE
:RESTORE *T
```

FILES RESTORED = 5

FILE	.GROUP	.ACCOUNT	LDN	ADDRESS
BASIC	.PUB	.SYS	2	%000000000044
FORTRAN	.PUB	.SYS	2	%000000004564
SPL	.PUB	.SYS	2	%000000006704
STAR	.PUB	.SYS	2	%000000007405
TAPECOPY	.PUB	.SYS	2	%000000007634

FILES NOT RESTORED = 0

```
:EOJ
MON, APR 2, 1973, 8:32 AM
```

```
:HELLO      username:account
SESSION NUMBER = #57
MON, APR  2, 1973,  8:32 AM
HP32000A.01.02
:FILE LIST;DEV=LP
:FILE T;DEV=TAPE
:STOPE STAR,BASIC;*T
:RESTORE *T;BASIC
:BYE
MON, APR  2, 1973,  8:35 AM
END OF SESSION
```

*listing on
LP
EVEN THOUGH
REQUEST BY
SESSION.*

on LP.

FILES STORED = 2

FILE	.GROUP	.ACCOUNT	LDN	ADDRESS
STAR	.PUB	.SYS	2	%000000007405
BASIC	.PUB	.SYS	2	%000000000044

FILES NOT STORED = 0

FILES RESTORED = 1

FILE	.GROUP	.ACCOUNT	LDN	ADDRESS
BASIC	.PUB	.SYS	2	%000000000044

FILES NOT RESTORED = 0

FOR ERROR CODE SEE SYSTEMS ANALYST NOTE NOVEMBER 16, 1972

3000 MPE. FILSYS-02

SYSDUMP

*↓ I/O configuration
dump*

.WHEN

FIRST TIME UP

AFTER CONFIGURATION CHANGES

AFTER CHANGES IN ACCOUNTS

.HOW

```
:HELLO username.account
SESSION NUMBER = #S1
TUE, MAY 8, 1973, 1:07 PM
HP32000A.01.06
:FILE TP;DEV=TAPE
:SYSDUMP *TP
```

ANY CHANGES?

END OF PROGRAM

:BYE

```
TUE, MAY 8, 1973, 1:10 PM
END OF SESSION
```

.TRY BY USING \$NULL ON TERMINAL

⌞ SYSDUMP \$NULL

ANY CHANGES?

REPLY TO

LOGICAL DEVICE NUMBER?

IS 0

↑ GET out of loop

SYSTEM DUMP → *HARDWARE PROGRAM*

. WHEN

SYSTEM CRASHED
SYSTEM HALT

. HOW

1. HALT

2. MAGNETIC TAPE

LOG. UNIT 7

KNOWN GOOD TAPE WITH WRITE RING, ON LINE, UNIT 0

3. SET ALL TOGGLE SWITCHES TO NORMAL

4. SET "B" SWITCH REGISTER TO 002006 OCTAL

5. PRESS I/O RESET

6. IF SYSTEM HALT LIGHT IS ON; RECORD STATUS
REGISTER THEN PRESS CPU RESET
OTHERWISE DO NOT PRESS

7. WATCH MAGNETIC TAPE FOR 8-15 FEET OF MOVEMENT
WHEN SYSTEM DUMP IS PRESSED.

8. PRESS SYSTEM DUMP

9. IF TRANSMISSION OF TAPE IS REQUESTED THEN

SET "B" REGISTER TO 006406

PRESS I/O RESET

PRESS CPU RESET

PRESS COLD LOAD TWICE

TAPE SHOULD MOVE EACH TIME

COLDLOAD IS PRESSED

*DOUBLE
EOP
MARK.*

. DISPOSITION

TAPE AND STATUS REGISTER VALUE
TO DATA CENTER SYSTEMS ANALYST

DEMO

JOB 1

```
:JOB      jobname,username.account,groupname
JOB NUMBER = #J1
FRI, MAR 30, 1973,  9:18 AM
HP32000A.01.02
:FORTRAN
```

+ SOURCE

:EOD - END OF DECK

```
END OF PROGRAM
:PREP $OLDPASS,$NEWPASS
:RUN $OLDPASS;STACK=3000
```

+ DATA

```
END OF PROGRAM
:E0J
FRI, MAR 30, 1973,  9:20 AM
```

JOB 2

```
:JOB      jobname,username.account,groupname
JOB NUMBER = #J2
FRI, MAR 30, 1973,  9:20 AM
HP32000A.01.02
:SPL
```

END OF DECK

:EOD

```
END OF PROGRAM
:PREP $OLDPASS,$NEWPASS
:PURGE TRANDEMO
:SAVE $OLDPASS,TRANDEMO
:E0J
```

```
FRI, MAR 30, 1973,  9:23 AM
```

*NOT
CONTINUE
NEEDED*

BATCH

```
:JOB      jobname,username.account
JOB NUMBER = #J3
FRI, MAR 30, 1973, 9:23 AM
HP32000A.01.02
:FILE MAST;DEV=TAPE
:SPL ,,,*MAST
```

COMPILE

:EOD

```
END OF PROGRAM
:EOJ
FRI, MAR 30, 1973, 9:32 AM
```

SESSION

```
:HELLO username.account
SESSION NUMBER = #S9
FRI, MAR 30, 1973, 9:51 AM
HP32000A.01.02
:RUN TRANDEMO
```

```
BASE
FT = 2
IN = 0
AREA
SQ IN = 48
SQ FT = 1
BASE
FT = /E
```

```
END OF PROGRAM
:BASIC
```

```
BASIC 1.0
>10 PRINT "HI THERE!"
>RUN
HI THERE!
```

>EXIT

```
END OF PROGRAM
:BYE
```

```
FRI, MAR 30, 1973, 9:53 AM
END OF SESSION
```

→ back TO TAPE

set throughout - JOB stream
 till next terminated

INPUT FILE

MASTER
 PLUS
 ADDITIONS

[, MASTER] [, NEW]
 \$NULL \$NULL

FILE SYSTEM COMMANDS

- Fortran [TEXT] [USE] [LIST]
 DEFAULT \$STDEN \$NEWPASS \$STDLIST
- PREP \$CLOPASS, \$NEWPASS
- RUN \$CLOPASS
- SAVE \$CLOPASS, \$CLOPASS
- FORT PREP = FORT + PREP
- FORT GO = FORT + PREP + GO.

FILE SYSTEM

3000 MPE COMMANDS

- PRECEDED BY COLON
- COMMAND NAME
- POSITIONAL PARAMETERS
- KEYWORD PARAMETERS
- CONTINUATION ALLOWED

:BUILD MASTERFL; TEMP; REC = -80,,F,ASCII

:PREP INPT,OUTP; &

:DL = 500 ; &

:CAP = PH,DS,MR

FORMAL DESIGNATOR - PROGRAM FILE NAME
(MAY BE ASSUMED)

ACTUAL FILE DESIGNATOR - ACTUAL FILE NAME

FILE COMMAND

- ☐ ESTABLISHES EQUATION BETWEEN FORMAL
AND ACTUAL DESIGNATORS
- ☐ RUN-TIME DESCRIPTION OF UNSPECIFIED FILE
CHARACTERISTICS
- ☐ RUN-TIME OVERRIDE OF FILE CHARACTERISTICS



FTN[~]xx

FILE COMMAND PARAMETERS

- 
- FORMAL DESIGNATOR (REQ) *ONLY REQUIRED PAR.*
 - ACTUAL FILE REFERENCE ,NEW/OLD
 - DEVICE TYPE
 - FILE SIZE
 - NUMBER OF EXTENTS
 - EXTENTS INITIALLY ALLOCATED
 - RECORD SIZE
 - BLOCKING FACTOR
 - RECORD TYPE

FILE COMMAND PARAMETERS

- ☐ CARRIAGE CONTROL OPTION
- ☐ CONCURRENT ACCESS
- ☐ MULT-RECORD I/O - *over record boundary writing*
- ☐ ACCESS TYPE
- ☐ NUMBER OF BUFFERS
- ☐ FILE TYPE
- ☐ FILE CODE
- ☐ BACK REFERENCED FILES

FILE COMMAND

DEVICE PARAMETER

;DEV = LP

CARD

TTY

TAPE1

DISC (DEFAULT)

FILE COMMAND

DISC PARAMETER

;DISC = FILE SIZE

NUMBER OF EXTENTS,

16 EXTENTS

INITIAL EXTENT ALLOCATION

NEEDED NOW

LOGICAL RECORDS



;DISC = 10000, 10, 5

FILE COMMAND

RECORD PARAMETER

128 words/sector

get units
boundaries
↓

;REC = RECORD SIZE, - BYTE
+ words

buffer
size

BLOCKING FACTOR,

records/sector
if omitted picks optimum

FIXED/UNDEFINED/VARIABLE LENGTH

pad with
zeros

BINARY/ASCII
default

padding
"BLANK" } also needed with editor

REC = -80,3,F,ASCII

*depending on
device type
default*

FILE COMMAND ACCESS PARAMETER

ACC = **IN** - read only

OUT - write only - ALL PROGRAM DATA DELETE

OUTKEEP - WRITE ONLY - SAVE ALL DATA

APPEND - WRITE AT END OF DATA

INOUT

UPDATE - ALL FILE IN PROGRESS

TRANSIENT
PARAMETER } NO PERMANENT

FILE COMMAND

MISCELLANEOUS PARAMETERS

; NOBUF — NO I/O BUFFERING

EXCLUSIVE ACCESS } ; CCTL — carriage control characters. 1st char = ccc

; EXC/EAR/SHR } ACCESS TYPE } read+write } CONCURRENTLY

; MR — Read only

; MR — MULTIRECORD ACCESS — read, write on record boundaries

; BUF=

AT FILE CLOSE TIME } ; DEL/SAVE/TEMP

; CODE=

FILE { USL. 1024

PROG. 1024

↑ KEPT UNTIL JOB OR SESSION TERMINATE THEN DELETE

FILE COMMAND
BACK-REFERENCE

- ☐ POINTS TO PREVIOUS FILE COMMAND
- ☐ TRANSFERS FILE CHARACTERISTICS
- ☐ ASTERISK INDICATES BACK-REFERENCE
- ☐ NO OTHER PARAMETERS ALLOWED

:FILE NEWFILE =*DUMMY

TOTAL Format

BACK REFERENCE EXAMPLES

:FILE MASTER; DEV=TAPE; REC=-100, 1, V, ASCII; &

: NOBUF; ACC=OUT; SAVE

:FILE TPFL1 = *MASTER

:FILE TPFL2 = *MASTER

:FILE TPFL3 = *MASTER

:FILE DUMMY; DEV=LP

:FILE SOURCE; DEV=CARD

:SPL *SOURCE,,*DUMMY

MPE FILE COMMAND

BUILD *on spot*

- ☐ ALLOCATES DISC SPACE
- ☐ FILE INITIALLY EMPTY
- ☐ IMMEDIATE ALLOCATION
- ☒ PERMANENT UNLESS TEMP INDICATED

MPE FILE COMMAND
BUILD

- ☐ FILE REFERENCE *ACTUAL FILE NAME*
- ☐ DEVICE TYPE *TYPICALLY DISC*
- ☐ DISC CHARACTERISTICS *SAME AS FILE COMMAND*
- ☐ RECORD CHARACTERISTICS *SAME AS FILE COMMAND*
- ☐ CARRIAGE CONTROL
- ☐ TEMPORARY
- ☐ TYPE CODE

:BUILD PROGMM; CODE = 1029

*↑ PROGRAM FROM SEGMENTER
1024 CODE FROM COMPILER
SOURCE ~~FROM~~ CODE NOT
NEEDED*

MPE FILE COMMANDS

: SAVE \$OLDPASS, PFILE

: SAVE PFILE

*ONLY
1 per PURGE*
: PURGE PFILE, TEMP

*IF TEMP, FILE
MUST SAY TEMP.*

: RENAME OLDFNM, NEWFNM, TEMP

: RESET DUMMY

KILLS ANY FILE COMMAND FOR OUTPUT

: RESET @

KILLS ALL FILE COMMANDS

: LISTF *FILE NAME* 1,2

DEVICE TYPE
FILE CHAIR → *1 SOME INFO.*
2 ALL INFO.

STAND ALONE DATA only
MPE FILE COMMAND
DATA — TEMPORARY only

- TO INPUT DATA SEPARATE FROM COMMAND STREAM
- CAN ACCESS FROM BATCH DEVICE WHILE IN SESSION MODE
- NOT FOR LABELED TAPES OR DISC FILES
- DATA CAN BE REFERENCED WITH A BACK-REFERENCE
- DATA DECK ENDS WITH :EOD

TAPE STATION
CARD reader
only

USERNAME, ACCOUNT; NAME CARD DECK
LOG ON

deck
in
card
reader
{
:DATA S9.UT; XXX
deck
: EOD
} FILE NAME

* FILE LIST, DEV=LP
* RUN LIST EQ .PUB. SYS

HP 3000

TEXT EDITOR

stan shell

RANGE EXPRESSIONS



LIST ALL

1 THIS IS LINE ONE*
2 THIS IS LINE TWO*
3 THIS IS LINE THREE*
4 THIS IS LINE FOUR*

ALL

LIST FIRST

1 THIS IS LINE ONE*

FIRST

LIST LAST

4 THIS IS LINE FOUR*

LAST

LIST *

4 THIS IS LINE FOUR*

*

LIST LAST-1

3 THIS IS LINE THREE*

LAST-1

LIST FIRST+2

3 THIS IS LINE THREE*

FIRST + 2

RANGE EXPRESSIONS

```
/ADD  
1   THIS IS THE FIRST LINE  
2   THIS IS THE SECOND LINE  
3   THIS IS THE THIRD LINE  
4   THIS IS THE LAST LINE  
5   //
```

...

```
/FIND FIRST  
1   THIS IS THE FIRST LINE  
    ↑
```

*/"THIRD"

```
/DELETE */"THIRD"
```

1

2

3

```
/LIST ALL
```

3 LINE

4 THIS IS THE LAST LINE

```
/FIND FIRST
```

3 LINE

↑

```
/DELETE */*+1
```

/+1

3

4

```
/LIST ALL
```

/

RANGE EXPRESSIONS

LIST ALL

1	THIS IS LINE ONE*
2	THIS IS LINE TWO*
3	THIS IS LINE THREE*
4	THIS IS LINE FOUR*

FIND 3

3	THIS IS LINE THREE*
---	---------------------

DELETE */*(5)

3

/(5)

ABSOLUTE COLUMN

LIST 3

3	IS LINE THREE*
---	----------------

FIND 4

4	THIS IS LINE FOUR*
---	--------------------

DELETE */*(+5)

4

/(+5)

LIST 4

4	LINE FOUR*
---	------------

TEXT EDITOR
TEXT INPUT COMMANDS

0

0

0

0

1

TEXT COMMAND

HELLO S9.UT,69
SESSION NUMBER = #53
WED, MAY 2, 1973, 11:11 AM
HP32000A.01.06
:

TEXT INPUT FROM
CARD READER

: FILE XXX;DEV=CARD;REC=40,,,ASCII

FILE COMMAND

: EDITOR

HP32201A.01.3 EDIT/3000 WED, MAY 2, 1973, 11:14 AM
/

TEXT *XXX,UNN

BACK-REFERENCE-UNNUMBERED

/ LIST ALL

1 THIS IS LINE ONE
2 THIS IS LINE TWO
3 THIS IS LINE THREE
4 THIS IS LINE FOUR
/

TEXT LONGF

IF IT IS OK TO CLEAR RESPOND "YES"
CLEAR? Y
/

TEXT WILL CLEAR BUFFER

LIST ALL

```

1  $CONTROL USLINIT
2      BEGIN  << START OF SPL PROGRAM>>
3      ARRAY BUFFER (0:79);
4      ARRAY FNAME (0:6):="MASTER ";
5      INTEGER FN;
6      INTRINSIC FOPEN,FCLOSE;
7      EQUATE PROPT=1
8      OPENFL: FN:=FOPEN(FNAME,1,0)<<OPEN THE FILE>>;
9      FCLOSE (FN,1,0)<<CLOSE THE FILE>>;
10     END.
/
```

JOIN COMMAND

LIST ALL

TEXT INPUT FROM DISC FILE

JOIN LONGF

JOIN WILL LIST

```

1      $CONTROL USLINIT
2      BEGIN  << START OF SPL PROGRAM>>
3      ARRAY BUFFER (0:79);
4      ARRAY FDNAME (0:6):="MASTER ";
5      INTEGER FN;
6      INTRINSIC FOPEN,FCLOSE;
7      EQUATE PROPT=1
8      OPENFL: FN:=FOPEN(FDNAME,1,0)<<OPEN THE FILE>>;
9      FCLOSE (FN,1,0)<<CLOSE THE FILE>>;
10     END.

```

JOIN SHORTE

JOIN WILL NOT CLEAR BUFFER

```

11     THIS IS LINE ONE
12     THIS IS LINE TWO
13     THIS IS LINE THREE
14     THIS IS LINE FOUR

```

LIST ALL

```

1      $CONTROL USLINIT
2      BEGIN  << START OF SPL PROGRAM>>
3      ARRAY BUFFER (0:79);
4      ARRAY FDNAME (0:6):="MASTER ";
5      INTEGER FN;
6      INTRINSIC FOPEN,FCLOSE;
7      EQUATE PROPT=1
8      OPENFL: FN:=FOPEN(FDNAME,1,0)<<OPEN THE FILE>>;
9      FCLOSE (FN,1,0)<<CLOSE THE FILE>>;
10     END.
11     THIS IS LINE ONE
12     THIS IS LINE TWO
13     THIS IS LINE THREE
14     THIS IS LINE FOUR

```

ADD COMMAND

ADD

1 THIS IS AN EXAMPLE
2 OF DATA INPUT THROUGH
3 THE ADD COMMAND OF
4 THE HP 300 TEXT EDITOR.
5 // WILL TERMINATE THE
6 ADD COMMAND ONLY IF
7 IT ENDS THE LINE
8 //

...
/

IF EMPTY BUFFER STARTS

AT LINE ONE

TERMINATES ADD

ADD WILL APPEND TO END OF

BUFFER

ADD

8 ADD WILL DEFAULT THE
9 LINE NUMBER TO THE END OF THE TEXT BUFFER
10 //

...
/

ADD 5.01

5.01 THIS IS HOW
5.02 YOU CAN SPECIFY
5.03 A LINE INCREMENT
5.04 LESS THAN 1.
5.05 IT
5.06 WILL
5.07 AUTOMATICALLY
5.08 TAKE
5.09 THE
5.1 NEW
5.11 INCREMENT
5.12 SPECIFIED IN THE COMMAND ITSELF.
5.13 //

...
/

CAN IMPLY A LINE INCREMENT

ADD 5.01

*15*COMMAND WILL NOT REPLACE OR INTERLEAVE LINES

...
/

ADD WILL NOT REPLACE LINES

INSERT COMMAND

/JOIN SHORT

```
1  THIS IS LINE ONE
2  THIS IS LINE TWO
3  THIS IS LINE THREE
4  THIS IS LINE FOUR
```

CAN INSERT WHOLE LINES

INSERT 2

```
2  THIS IS LINE TWO
   NOW THIS IS THE NEW LINE TWO
```

```
2.1 AND THIS WILL BE 2.1
```

```
2.2 //
```

```
THIS IS LINE TWO
```

PRE-EXISTING LINES WILL

SLIP DOWN IN SEQUENCE

LIST ALL

```
1  THIS IS LINE ONE
2  NOW THIS IS THE NEW LINE TWO
2.1 AND THIS WILL BE 2.1
2.2 THIS IS LINE TWO
3  THIS IS LINE THREE
4  THIS IS LINE FOUR
```

INSERT 4(9)

//WILL KEEP REST OF LINE

```
4  THIS IS LINE FOUR
```

```
THE LINE AFTER THREE. IT IS //
```

```
LINE FOUR
```

/LIST 4

```
4  THIS IS THE LINE AFTER THREE. IT IS LINE FOUR
```

INSERT COMMAND

JOIN SHORTF

1 THIS IS LINE ONE
2 THIS IS LINE TWO
3 THIS IS LINE THREE
4 THIS IS LINE FOUR

/

INSERT 4(+4)

4 THIS IS LINE FOUR
*NOW //
IS LINE FOUR

/

LIST 4

4 THIS NOW IS LINE FOUR

/

INSERT 4(6)

4 THIS NOW IS LINE FOUR

5

//

NOW IS LINE FOUR

/LIST ALL

1 THIS IS LINE ONE
2 THIS IS LINE TWO
3 THIS IS LINE THREE
4 THIS
5 NOW IS LINE FOUR

/

INSERT AT 4TH NON BLANK

CHARACTER

INSERT AT 6TH COLUMN POSITION

CARRIAGE RETURN WILL SPLIT LINE

TEXT EDITOR

TEXT MANIPULATION COMMANDS

 CHANGE

 DELETE

 REPLACE

 GATHER

 MODIFY

CHANGE COMMAND

/ADD

```
1  THIS IS LINE ONE
2  THIS IS LINE TWO
3  THIS IS LINE THREE
4  THIS IS LINE FOUR
5  //
```

...

/CHANGE "ONE" TO "1" IN 1

```
1  THIS IS LINE 1
```

CHANGE A STRING IN A LINE

CHANGE 1/1 TO "WHILE T" IN 2

```
2  WHILE THIS IS LINE TWO
```

/CHANGE 1 TO " " IN 3

```
3  THIS IS LINE THREE
```

/CHANGE "LINE" TO "ITEM" IN ALL

```
1  THIS IS ITEM 1
2  WHILE THIS IS ITEM TWO
3  THIS IS ITEM THREE
4  THIS IS ITEM FOUR
```

CHANGE A COLUMN

PREFIX A LINE

CHANGE ALL STRINGS IN A

RANGE

DELETE COMMAND

/JOIN LONGF

```

1      $CONTROL USLINIT
2      BEGIN << START OF SPL PROGRAM>>
3      ARRAY BUFFER (0:79);
4      ARRAY FDNAME (0:6):="MASTER ";
5      INTEGER FN;
6      INTRINSIC FOPEN,FCLOSE;
7      EQUATE PROPT=1
8      OPENFL: FN:=FOPEN(FDNAME,1,0)<<OPEN THE FILE>>;
9      FCLOSE (FN,1,0)<<CLOSE THE FILE>>;
10     END.

```

/DELETE 3/4

DELETE RANGE OF LINES

3
4

/LIST ALL

```

1      $CONTROL USLINIT
2      BEGIN << START OF SPL PROGRAM>>
5      INTEGER FN;
6      INTRINSIC FOPEN,FCLOSE;
7      EQUATE PROPT=1
8      OPENFL: FN:=FOPEN(FDNAME,1,0)<<OPEN THE FILE>>;
9      FCLOSE (FN,1,0)<<CLOSE THE FILE>>;
10     END.

```

/DELETE 2(15)/6(20)

DELETE COLUMNS IN MULTIPLE LINES

2
5
6

/LIST ALL

```

1      $CONTROL USLINIT
2      BEGIN
6      OPEN,FCLOSE;
7      EQUATE PROPT=1
8      OPENFL: FN:=FOPEN(FDNAME,1,0)<<OPEN THE FILE>>;
9      FCLOSE (FN,1,0)<<CLOSE THE FILE>>;
10     END.

```

/DELETE 7/8,10

DELETE COMPLEX RANGE

7
8
10

LIST ALL

```

1      $CONTROL USLINIT
2      BEGIN
6      OPEN,FCLOSE;
9      FCLOSE (FN,1,0)<<CLOSE THE FILE>>;

```

REPLACE COMMAND

#/JOIN SHORT

```
1  THIS IS LINE ONE
2  THIS IS LINE TWO
3  THIS IS LINE THREE
4  THIS IS LINE FOUR
```

/REPLACE 3

REPLACE COMPLETE LINE

```
3  THIS IS LINE THREE
3  NOW THIS IS A WHOLE NEW LINE NUMBER THREE
```

/LIST ALL

```
1  THIS IS LINE ONE
2  THIS IS LINE TWO
3  NOW THIS IS A WHOLE NEW LINE NUMBER THREE
4  THIS IS LINE FOUR
```

/REPLACE 1/2

REPLACE A RANGE OF LINES

```
1  THIS IS LINE ONE
1  *****LINE ONE*****
2  THIS IS LINE TWO
2  *****TWO*****
```

/LIST ALL

```
1  *****LINE ONE*****
2  *****TWO*****
3  NOW THIS IS A WHOLE NEW LINE NUMBER THREE
4  THIS IS LINE FOUR
```

/REPLACE 1,3/4

REPLACE COMPLEX RANGE

```
1  *****LINE ONE*****
1  $$$$11111$$$$$
3  NOW THIS IS A WHOLE NEW LINE NUMBER THREE
3  $$$33333$$$$$
4  THIS IS LINE FOUR
4  $$$44444$$$$$
```

/LIST ALL

```
1  $$$11111$$$$$
2  *****TWO*****
3  $$$33333$$$$$
4  $$$44444$$$$$
```

GATHER COMMAND

/JOIN LONGF

```

1 $CONTROL USLINIT
2 BEGIN << START OF SPL PROGRAM>>
3 ARRAY BUFFER (0:79);
4 ARRAY FDNAME (0:6):="MASTER ";
5 INTEGER FN;
6 INTRINSIC FOPEN,FCLOSE;
7 EQUATE PROPT=1
8 OPENFL: FN:=FOPEN(FDNAME,1,0)<<OPEN THE FILE>>;
9 FCLOSE (FN,1,0)<<CLOSE THE FILE>>;
10 END.
```

/GATHER 3/4 TO 5.001

MOVE RANGE TO NEW RANGE WITH NEW

```

3 := 5.001
4 := 5.002
```

/LIST ALL

INCREMENT

```

1 $CONTROL USLINIT
2 BEGIN << START OF SPL PROGRAM>>
5 INTEGER FN;
5.001 ARRAY BUFFER (0:79);
5.002 ARRAY FDNAME (0:6):="MASTER ";
6 INTRINSIC FOPEN,FCLOSE;
7 EQUATE PROPT=1
8 OPENFL: FN:=FOPEN(FDNAME,1,0)<<OPEN THE FILE>>;
9 FCLOSE (FN,1,0)<<CLOSE THE FILE>>;
10 END.
```

/GATHER 5/LAST TO 500

RANGE INCREMENT DEFAULTS TO ONE

```

5 := 500
5.001 := 501
5.002 := 502
6 := 503
7 := 504
8 := 505
9 := 506
10 := 507
```

/LIST ALL

```

1 $CONTROL USLINIT
2 BEGIN << START OF SPL PROGRAM>>
500 INTEGER FN;
501 ARRAY BUFFER (0:79);
502 ARRAY FDNAME (0:6):="MASTER ";
503 INTRINSIC FOPEN,FCLOSE;
504 EQUATE PROPT=1
505 OPENFL: FN:=FOPEN(FDNAME,1,0)<<OPEN THE FILE>>;
506 FCLOSE (FN,1,0)<<CLOSE THE FILE>>;
507 END.
```

/GATHER ALL

GATHER ALL WILL RENUMBER TEXT

/LIST ALL

```

1 $CONTROL USLINIT
2 BEGIN << START OF SPL PROGRAM>>
3 INTEGER FN;
4 ARRAY BUFFER (0:79);
5 ARRAY FDNAME (0:6):="MASTER ";
6 INTRINSIC FOPEN,FCLOSE;
7 EQUATE PROPT=1
8 OPENFL: FN:=FOPEN(FDNAME,1,0)<<OPEN THE FILE>>;
9 FCLOSE (FN,1,0)<<CLOSE THE FILE>>;
10 END.
```

MODIFY COMMAND



/LIST ALL

```

1  SCONTROL USLINIT
2      BEGIN  <<START OF SPL PROGRAM>>
3      ARRAY BUFFER (0:79);
4      ARRAY MESSAGE (0:10);
5      INTEGER DSCFN;
6      EQUATE PROPT=1
7  CLOSEFL: FCLOSE (DSCFN,1,0);<<MAKE A PERMANENT FILE>>
8      IF <> THEN PRINTOP(MESSAGE,-22,0);
9      END
    
```

/MODIFY 2

"D" WILL DELETE CHARACTER

MODIFY 2

```

      BEGIN  <<START OF SPL PROGRAM>>
    
```

```

      BEGIN  <<START OF DDD PROGRAM>>
    
```

MULTIPLE "D" WILL BRACKET

```

      BEGIN  D D
    
```

DELETE RANGE

/MODIFY 4

MODIFY 4

```

      ARRAY MESSAGE (0:10);
    
```

```

      ARRAY MESSAGE (0:10):=I:="FCLOSE ERROR";
    
```

"I" WILL INSERT STRING

/MODIFY 7

MODIFY 7

```

      CLOSEFL: FCLOSE (DSCFN,1,0);<<MAKE A PERMANENT FILE>>
    
```

RCLSFILE

```

      CLSFILE: FCLOSE (DSCFN,1,0);<<MAKE A PERMANENT FILE>>
    
```

"R" WILL REPLACE CHARACTERS

MODIFY COMMAND (CONTINUED)

/MODIFY 8 //WILL RESTORE ORIGINAL LINE CONTENTS

```
MODIFY      8
            IF <> THEN PRINTOP(MESSAGE,-22,0);
            R>      I ***THIS IS AN ERROR***
            IF >      I ***THIS IS AN ERROR***;
```

```
//
MODIFY      8
            IF <> THEN PRINTOP(MESSAGE,-22,0);
```

/MODIFY 2/4 MODIFY A RANGE OF LINES

```
MODIFY      2
            BEGIN
```

```
IA:
A:          BEGIN
```

```
MODIFY      3
            ARRAY BUFFER (0:79);
```

```
IB:
B:          ARRAY BUFFER (0:79);
\MODIFY    4
            ARRAY MESSAGE (0:10):="FCLOSE ERROR";
```

```
IC:
C:          ARRAY MESSAGE (0:10):="FCLOSE ERROR";
```

/

LIST ALL

```
1      $CONTROL USLINIT
2      A:          BEGIN
3      B:          ARRAY BUFFER (0:79);
4      C:          ARRAY MESSAGE (0:10):="FCLOSE ERROR";
5
6      INTEGER DSCFN;
7      EQUATE PROPT=1
8      CLSFILE: FCLOSE (DSCFN,1,0);<<MAKE A PERMANENT FILE>>
9      IF <> THEN PRINTOP(MESSAGE,-22,0);
END
```

MODIFY COMMAND (CONTINUED)

```

/ MODIFY 8                                // WILL RESTORE ORIGINAL LINE CONTENTS
MODIFY      8
            IF <> THEN PRINTOP(MESSAGE,-22,0);
            R>      I ***THIS IS AN ERROR***
            IF >      I ***THIS IS AN ERROR***;

//
MODIFY      8
            IF <> THEN PRINTOP(MESSAGE,-22,0);

```

```

/ MODIFY 2/4                                MODIFY A RANGE OF LINES
MODIFY      2
            BEGIN

IA:
A:          BEGIN

MODIFY      3
            ARRAY BUFFER (0:79);

```

```

IB:
B:          ARRAY BUFFER (0:79);
\MODIFY      4
            ARRAY MESSAGE (0:10):="FCLOSE ERROR";

IC:
C:          ARRAY MESSAGE (0:10):="FCLOSE ERROR";

/

```

```

/ LIST ALL
1      $CONTROL USLINIT
2      A:          BEGIN
3      B:          ARRAY BUFFER (0:79);
4      C:          ARRAY MESSAGE (0:10):="FCLOSE ERROR";
5
6      INTEGER DSCFN;
7      EQUATE PROPT=1
8      CLSFILE: FCLOSE (DSCFN,1,0);<<MAKE A PERMANENT FILE>>
9      IF <> THEN PRINTOP(MESSAGE,-22,0);
END

```

TEXT EDITOR
TEXT OUTPUT COMMANDS

● KEEP

KEEP DUMMYFL

WRITES BUFFER TO NEW DISC FILE

/KEEP DUMMYFL

DUMMYFL ALREADY EXISTS - RESPOND YES TO PURGE OLD AND THEN KEEP
PURGE OLD?Y

WILL WARN IF FILE ALREADY EXISTS

● LIST

/LIST ALL

1 THIS IS LINE ONE
2 THIS IS LINE TWO
3 THIS IS LINE THREE
4 THIS IS LINE FOUR

LIST ALL OFFLINE

*OFFLINE LIST DEVICE NOT AVAILABLE

/LIST ALL OFFLINE

LIST A RANGE ON PRINTER

TEXT EDITOR
MISCELLANEOUS COMMANDS

☐ HOLD

☐ FIND

☐ SET

☐ VERIFY

☐ END

HOLD COMMAND/FILE

/ADD

1 <<<<<HOLD FILE CONTENTS --LINE ONE>>>>>
2 <<<<<HOLD FILE CONTENTS --LINE TWO>>>>>

WRITES RANGE TO HOLD FILE

/HOLD 1/2

<<<<<HOLD FILE CONTENTS --LINE ONE>>>>>
<<<<<HOLD FILE CONTENTS --LINE TWO>>>>>

/TEXT SHORTF

/ADD ,HOLD

CAN ADD HOLD FILE

5 THIS LINE IS INPUT FROM THE TERMINAL

6 <<<<<HOLD FILE CONTENTS --LINE ONE>>>>> CONTENTS AFTER OTHER
7 <<<<<HOLD FILE CONTENTS --LINE TWO>>>>> LINE INPUT

/LIST ALL

1 THIS IS LINE ONE
2 THIS IS LINE TWO
3 THIS IS LINE THREE
4 THIS IS LINE FOUR
5 THIS LINE IS INPUT FROM THE TERMINAL

6 <<<<<HOLD FILE CONTENTS --LINE ONE>>>>> IMMEDIATELY ADDS
7 <<<<<HOLD FILE CONTENTS --LINE TWO>>>>>

/ADD ,HOLD,NOW

CONTENTS OF HOLD FILE

8 <<<<<HOLD FILE CONTENTS --LINE ONE>>>>>
9 <<<<<HOLD FILE CONTENTS --LINE TWO>>>>>

/LIST ALL

1 THIS IS LINE ONE
2 THIS IS LINE TWO
3 THIS IS LINE THREE
4 THIS IS LINE FOUR
5 THIS LINE IS INPUT FROM THE TERMINAL
6 <<<<<HOLD FILE CONTENTS --LINE ONE>>>>>
7 <<<<<HOLD FILE CONTENTS --LINE TWO>>>>>
8 <<<<<HOLD FILE CONTENTS --LINE ONE>>>>>
9 <<<<<HOLD FILE CONTENTS --LINE TWO>>>>>

/DELETE ALL

/HOLD;

/ADD 1,HOLD

WARNING - HOLD IS NULL

HOLD WILL CLEAR HOLD FILE

//

/LIST ALL

FIND COMMAND

/JOIN SHORTF

1 THIS IS LINE ONE
2 THIS IS LINE TWO
3 THIS IS LINE THREE
4 THIS IS LINE FOUR

FIND A LINE

/FIND 4

4 THIS IS LINE FOUR

FIND CURRENT LINE

/FIND *

4 THIS IS LINE FOUR

FIND FIRST LINE

/FIND FIRST

1 THIS IS LINE ONE

/FIND "TWO"

2 THIS IS LINE TWO

FIND A STRING

/FIND "ONE"

*21*STRING NOT FOUND BEFORE LIMIT FIND WILL SEARCH FROM

--- POINTER TO END OF BUFFER

/FIND FIRST

1 THIS IS LINE ONE

/FIND "ONE"/*(+12)

1 THIS IS LINE ONE

SEARCH TO 12TH NON-

BLANK CHARACTERS

/FIND FIRST:

1 THIS IS LINE ONE

SEARCH TO 12TH CHARACTER

/FIND "ONE"/*(12)

*21*STRING NOT FOUND BEFORE LIMIT

/FIND FIRST

1 THIS IS LINE ONE

/FIND "THREE"/*+3

3 THIS IS LINE THREE

WILL SEARCH A RANGE OF
LINES

SET COMMAND

/SET RIGHT =10

RIGHT COLUMN POSITION SET

/SET SHORT

SHORT ALLOWS COMMAND ABBREVIATION

/A

1 ABCDEFGHIJKLMN0P
2 12345678901234567890
3 //

...

/L ALL

1 ABCDEFGHIJ
2 1234567890

NOTE ONLY 10 COLUMNS

/SET LONG

LONG REVERSES SET SHORT

/A

* 1*INVALID COMMAND NAME

/SET DELTA=10

CHANGE INCREMENT NUMBER

/ADD

12 THIS IS AN EXAMPLE
22 OF A DIFFERENT
32 LINE INCREMENT
42 SET BY DELTA//

...

/

/

/SET QUIET

INHIBIT EDITOR VERBALIZATION

/DELETE 22/LAST

/

//

/SET DISPLAY

/

/DELETE 1/12

1
2
12

/

VERIFY COMMAND

/ADD

1 LINE ONE
2 LINE TWO//

/VERIFY

2 LINE TWO
 ↑

VERIFY CURRENT POINTER

/VERIFY ALL

2 LINE TWO
 ↑

VERIFY ALL VARIABLES THAT CAN BE SET

POLL = TRUE (I.E. BATCH = FALSE)

REAR = TRUE (I.E. FRONT = FALSE)

DELTA = 1

CURRENT DEPTH = 0, THE DEPTH LIMIT = 10

RIGHT = 72

LENGTH = 72

LONG = TRUE (I.E. SHORT = FALSE)

TIME = 50

TOTAL NUMBER OF CURRENT LINES = 2

FROM = 1

LEFT = 1

FIXED = TRUE (I.E. VARIABLE = FALSE)

SIZE = 0

DISPLAY = TRUE (I.E. QUIET = FALSE)

END COMMAND

END

IF IT IS OK TO CLEAR RESPOND "YES"

CLEAR? YES

END OF PROGRAM

:

TERMINATES

TEXT EDITOR

BASIC EDITOR
SUBSET FOR PROGRAM
DEVELOPMENT

- ☐ TEXT
- ☐ ADD
- ☐ KEEP
- ☐ LIST
- ☐ CHANGE
- ☐ DELETE
- ☐ MODIFY
- ☐ FIND
- ☐ END

MPE/3000

SEGMENTER



SUBSYSTEM

SEGMENTER TOPICS

- BACKGROUND INFORMATION
- MANAGING USL'S
- MANAGING LIBRARIES (OPTIONAL)
- LISTS & MAPS (OPTIONAL)

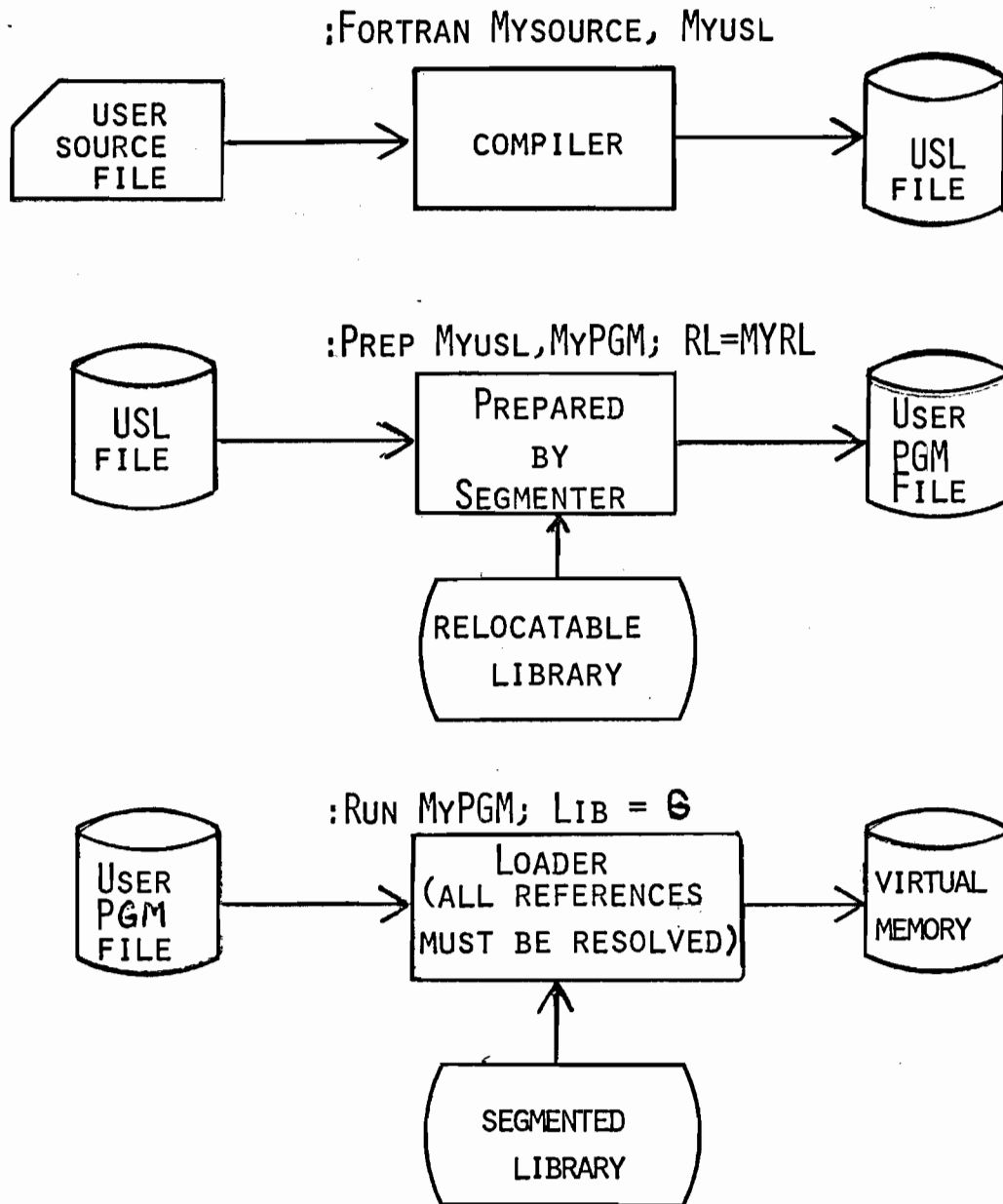
SEGMENTER FUNCTIONS

- 1. MANAGE USL'S: USER SUBPROGRAM LIBRARIES
- 2. MANAGE RL'S: RELOCATABLE LIBRARIES- CONTAINS PROCEDURES OR SUBROUTINES IN USL FORM.
- 3. MANAGE SL'S: SEGMENTED LIBRARIES: CONTAINS PROCEDURES OR SUBROUTINES THAT HAVE BEEN PREPARED

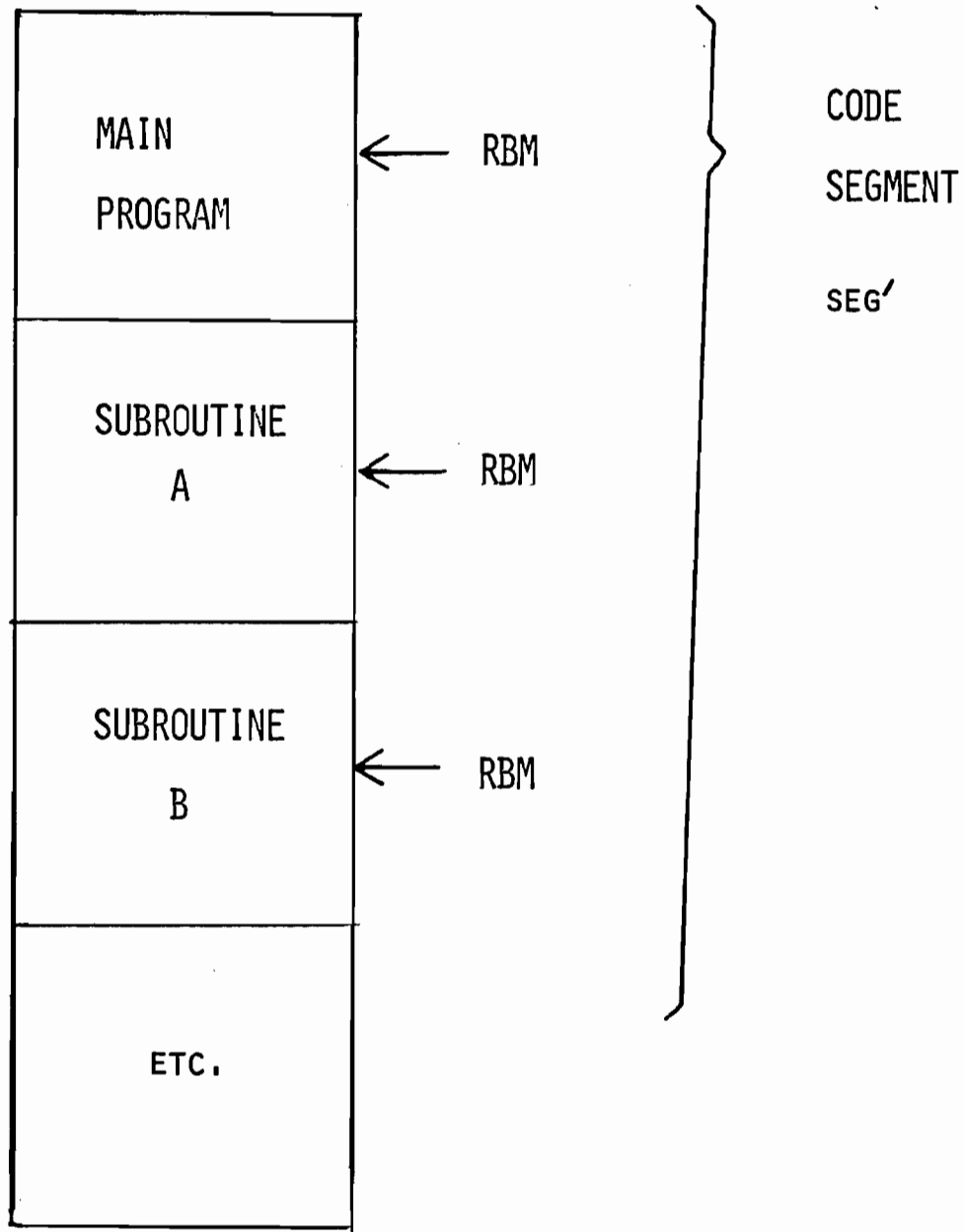
result of a compilation

result of preparation

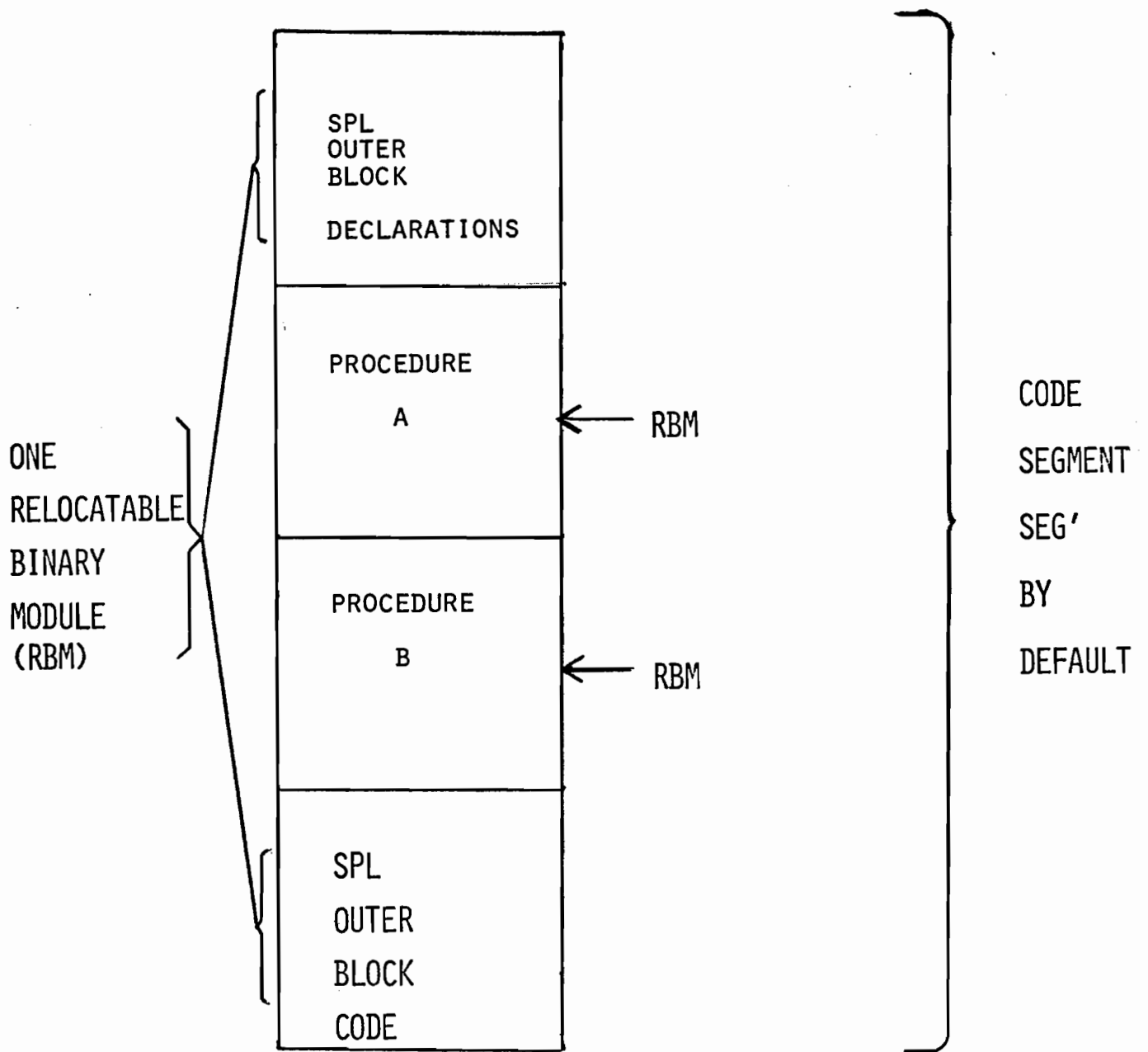
PROGRAM FLOW FROM SOURCE TO EXECUTION



RBM COMPONENTS OF A
FORTRAN PROGRAM



RBM COMPONENTS OF SPL PROGRAM



USL FILE COMPONENTS

- A USL CONSISTS OF ONE OR MORE SEGMENTS
- A SEGMENT CONTAINS ONE OR MORE RBM's
- AN RBM HAS ONE OR MORE ENTRY POINTS (ONE IN FORTRAN)

FILE LABEL & FILE INFO		
USL INFO		
CODE SEGMENT 1		
RBM1	RBM2	ETC.
CODE SEGMENT 2		

PROGRAM FILE (PF)

FILE LABEL & FILE INFO
PF INFO
CODE SEGMENT 1
CODE SEGMENT 2
. . . ETC.

REQUIRED

OPTIONAL

CONTROL OF SEGMENTATION AT COMPILE TIME

SPL PROGRAM

:SPL, MYUSL

BEGIN

[\$CONTROL MAIN = NAM1] <<RBMNAME, DEFAULT IS OB'>>]

INTEGER I,J,K;

REAL X,Y,Z;

INTRINSIC REAK, PRINT, ASCII;

[\$CONTROL SEGMENT=SEG2] <<DEFAULT - SEG'>>

REAL PROCEDURE ABC(X);

INTEGER X;

X:= Y*Z;

[\$CONTROL SEGMENT=SEG1] <<DEFAULT-NAME OF PREVIOUS SEGMENT>>

Z := REAL (I*J+K)*ABC(REAL(I**J)/Z); FIRST EXECUTABLE STATEMENT

END.<<END OF PROGRAM>>

CONTROL OF SEGMENTATION AT COMPILE TIME

FORTRAN PROGRAM

:FORTRAN,MYUSL

[\$CONTROL SEGMENT = SEG1]

PROGRAM NAM1

I = 1

STOP

END

[\$CONTROL SEGMENT = SEG 2]

FUNCTION ABC(X)

INTEGER X

ABC = X

RETURN

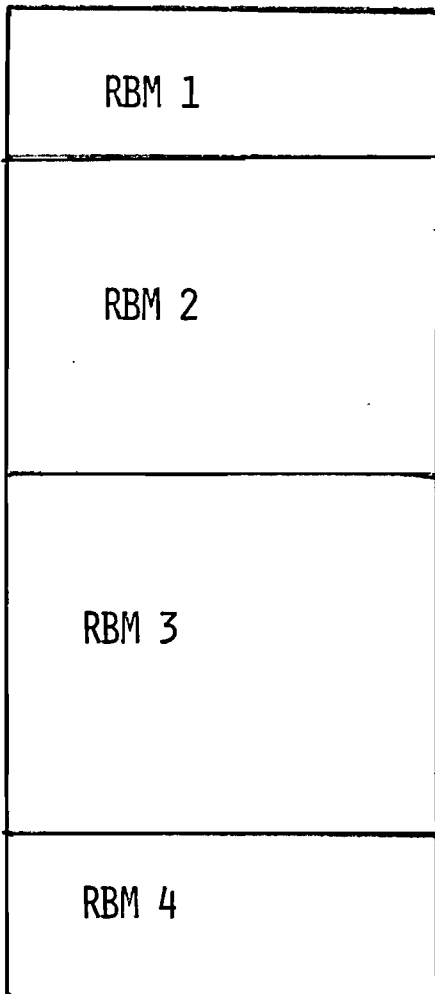
END

*SEG1
(default)*

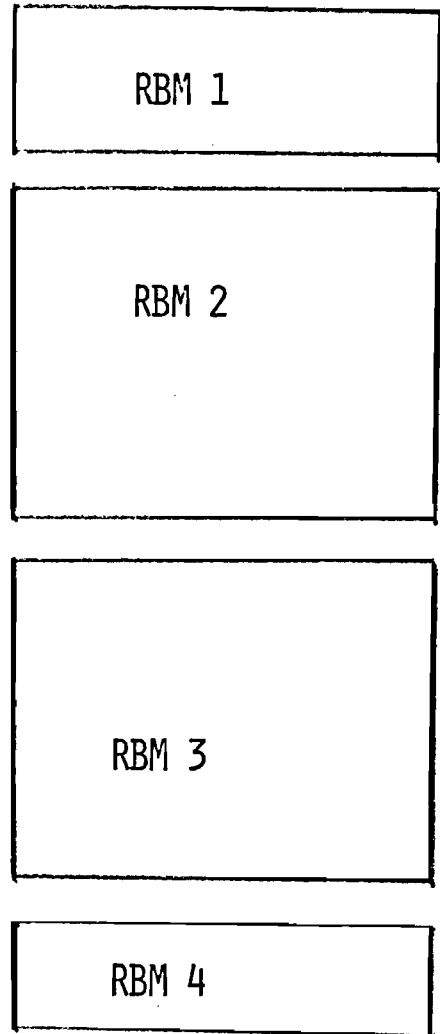
*main
(default)*

RESULTS OF SEGMENTATION

ONE SEGMENT



MANY SEGMENTS



ONE SEGMENT

NEEDS BIGGER CORE SLOT
MORE LIKELY TO BE OVERLAYED

SEVERAL SEGMENTS

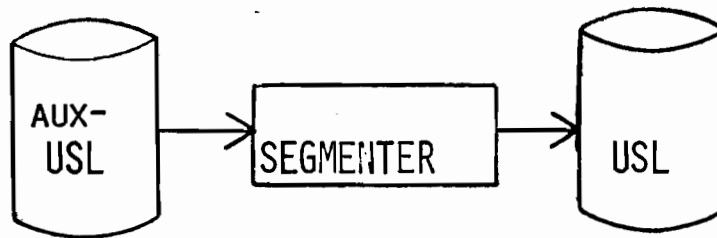
USES ADDITIONAL CST ENTRIES

INVOKING SEGMENTER

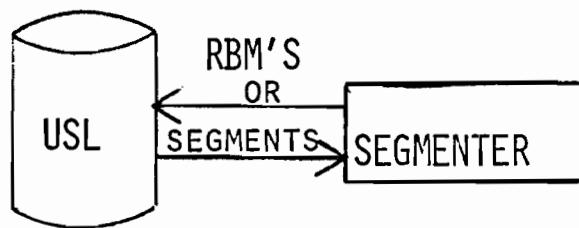
[FILE LP;DEV = LP]

:SEGMENTER [*LP]

SEGMENTER MANAGING OF USL'S



OR



SEGMENTER
USL MANAGING
COMMANDS

- BUILDUSL FILENAME, ETC.
- USL FILENAME
- CEASE
- USE
- NEWSEG
- PURGERBM
- LISTUSL
- AUXUSL
- COPY
-
— HIDE
-
— REVEAL

LISTUSL COMMAND

:SEGMENTER

-USL USLX

-LISTUSL



USL FILE USLX

SEG1 ← SEGMENT NAME

P1 } ENTRY POINT
P2 } NAMES
P3 }

SEG2

P4

PARTIAL

LISTING
OF

OUTPUT

FROM

LISTUSL

COMMAND



CONTROL OF SEGMENTATION VIA SEGMENTER

-NEWSEG JONES,SUSAN

-LISTUSL

USL FILE USLX. GROUP.ACCT

SEG1

P1

P2

SEG2

P3

P4

-EXIT

NEWSEG COMMAND

NEWSEG

NEWSEGNAME, RBMNAME (INDEX)

NEWSEGNAME

1 TO 15 CHARACTERS - MAY INCLUDE ANY ALPHANUMERIC OR SPECIAL CHARACTER EXCEPT ,;=&

THE NAME MAY BE A PREVIOUSLY EXISTING SEGMENT OR YOU MAY DEFINE A NEW ONE

RBMNAME

THE NAME OF ONE OF THE EXISTING RBM'S IN THE USL

INDEX

AN INTEGER SPECIFYING THE VERSION IF MORE THAN ONE VERSION OF THE RBM - LATEST IS 1, NEXT LATEST IS 2, ETC. IN ADDITION 0 OR NO INDEX IS THE MOST RECENTLY ACTIVATED AND STILL ACTIVE VERSION

CONTROL OF MULTIPLE VERSIONS

-USE -CEASE -PURGERBM	[ENTRY UNIT SEGMENT]	NAME	[(INDEX)]
-----------------------------	------------------------------	------	-------------

ENTRY, NAME - REFERS TO PRIMARY (RMBNAME) OR SECONDARY ENTRY POINT

UNIT, NAME - NAME IS THAT OF PRIMARY ENTRY POINT OF RBM AND REFERS TO IT AND ALL ENTRY POINTS WITHIN THE RBM.

SEGMENT, NAME- NAME IS THAT OF SEGMENT AND REFERS TO ALL ENTRY POINTS IN ALL RBM's WITHIN SEGMENT.

INDEX EXPLAINED ON PREVIOUS PAGE.

NOTE: FORTRAN ALLOWS ONLY ONE ENTRY POINT PER RBM

MULTIPLE VERSIONS OF RBM IN ONE USL

- HISTORY OF PROGRAM DEVELOPMENT CAN BE KEPT - LATEST VERSION,..., EARLIEST VERSION.
- OLD VERSIONS MAY BE ACTIVATED, DEACTIVATED, PURGED
- A "\$CONTROL USLINIT" CARD IN A COMPILATION "ERASES" THE CONTENTS OF A USL

MULTIPLE VERSION EXAMPLES

-USE ENTRY, EP1

ACTIVATES ENTRY POINT EP1, MOST RECENTLY ACTIVATED
VERSION

-CEASE UNIT, RBMX1(2)

DEACTIVATES ALL ENTRY POINTS WITHIN 2ND VERSION OF
RBMX1

-PURGERBM SEGMENT, SEG'(3)

"ERASES" 3RD MOST RECENTLY ACTIVATED VERSION OF
SEGMENT SEG' AND ALL ITS ENTRY POINTS

SEGMENTER PREPARE COMMAND

- THE SEGMENTER CAN PREPARE A USL INTO A PROGRAM FILE

-PREPARE PFNAME ;OPTIONS (REF MPE 7-12)

- NO USL IS MENTIONED IN THE PREPARE COMMAND. APPROPRIATE USL MUST BE INVOKED WITH USL COMMAND.

- IF PFNAME IS NOT AN OLD FILE A NEW TEMPORARY FILE IS SET UP.

- EG.
:SEGMENTER
-USL MYUSL
-PREPARE MYPGM
-EXIT

THE ABOVE SEQUENCE OF COMMANDS PREPS THE USL FILE,
MYUSL, INTO THE PROGRAM FILE MYPGM.

LIBRARIES

- THE SEGMENTER ALLOWS CONTROL OF TWO TYPES OF LIBRARIES
- RELOCATABLE LIBRARIES (RL's) ARE REFERRED TO AT PREPARATION TIME TO BE LINKED INTO USER'S PROGRAMS
- SEGMENTED LIBRARIES (SL's)

RELOCATABLE LIBRARY

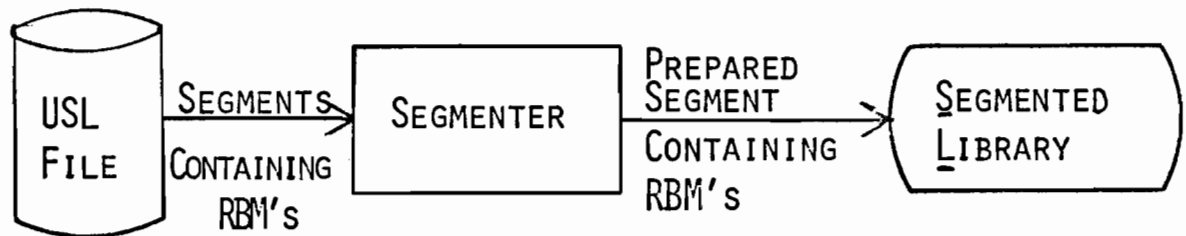
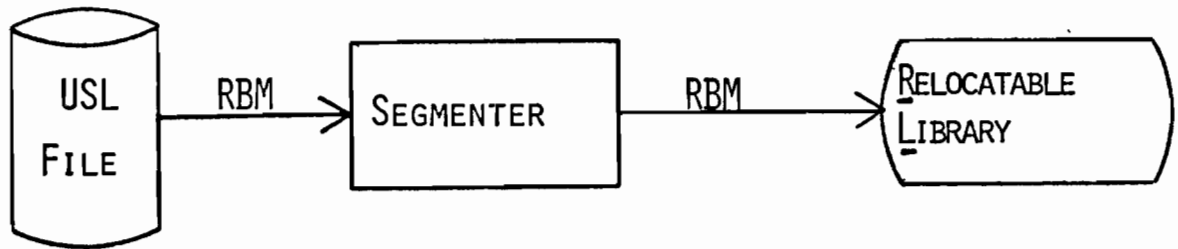
RL'S ARE COLLECTIONS OF RBM'S

(SUBROUTINES, PROCEDURES, MAIN PROGRAMS OR OUTER
BLOCKS) IN UNPREPARED USL FORM.

SEGMENTED LIBRARY

SL'S ARE COLLECTIONS OF RBM'S IN PREPARED PROGRAM
FORM.

LIBRARY MANAGEMENT



SEGMENTER
COMMANDS
FOR
MANAGING
LIBRARIES

RL'S

-BUILDRL FILENAME

-RL FILENAME *RL*

-ADDRL NAME

*↑ USE YOUR
OWN*

-PURGERL

TABLES

RBMS

-LISTRL

SL'S

-BUILDSL FILENAME

-SL FILENAME *SL*

-ADDSL NAME

*add seg into
whole
segment
on SL*

-PURGESL

-LISTSL

-HIDE

-REVEAL

BUILD SL COMMAND

- BUILDS A FILE WITH THE PROPER CHARACTERISTICS SO THAT IT MAY BE USED AS A SEGMENTED LIBRARY.
- EG. -BUILDSL SL.MYGROUP.MYACCT, 2000,5 BUILDS AN SL WITH A MAXIMUM OF 2000 128 WORD RECORDS, WHICH ARE GROUPED INTO FIVE EXTENTS.
- ONLY ONE SL MAY EXIST PER GROUP. ITS UNQUALIFIED NAME MUST BE "SL".

*RUN PUTS
all SEGMENTS
INTO VIRTUAL
MEMORY
BUT only first segment into
real memory*

SL COMMAND

- SPECIFIES A SEGMENTED LIBRARY TO BE USED AS THE OBJECT OF THE ADDSL, PURGESL, AND LISTSL COMMANDS.
- THE SL SPECIFIED MAY ONLY BE ONE OF 3 FORMS:
 1. THE SYSTEM LIBRARY SL
-SL SL. PUB. SYS
 2. THE ACCOUNT LIBRARY SL
-SL SL. PUB. MYACCT
 3. THE GROUP LIBRARY SL
-SL SL. MYGROUP. MYACCT

ADDSL COMMAND

- PREPARES A SEGMENT FROM THE USL (DESIGNATED BY THE USL COMMAND) AND MOVES IT INTO THE SEGMENT LIBRARY DESIGNATED BY THE SL COMMAND.

- EG
 - :SEGMENTER
 - USL MYUSL
 - SL SL
 - ADDSL MYSEG
 - EXIT

IN THE ABOVE SEQUENCE OF COMMANDS THE ADDSL COMMAND INSTRUCTS THE SEGMENTER TO PREPARE MYSEG, A SEGMENT LOCATED IN MYUSL AND PLACE THE PREPARED SEGMENT IN SL.

PURGESL COMMAND

- PURGES EITHER A SEGMENT OR AN ENTRY POINT FROM THE SL.
- EG. :SEGMENTER
-SL SL
-PURGESL SEGMENT,MYSEG
-EXIT
- THE ABOVE COMMANDS PURGE THE SEGMENT, MYSEG, FROM THE SL.

LISTS AND MAPS

:JOB MANAGER.SYS, FUB
JOB NUMBER = #J1
TUE, JAN 9, 1973, 7:07 AM
HP3200A.X1.26
:BUILD SEGU\$LCODE=1024
:BUILD SEGU\$LCODE=1029
:SPL SEGTEXT,SEGU\$LCODE=1024

```

00001000 00000 0 SCNTROL USLINIT
00002000 00000 0 BEGIN
00003000 00000 1 INTEGER I,J,K;
00004000 00000 1 REAL X,Y,Z;
00005000 00000 1 ARRAY ARY(0:15);
00006000 00000 1 INTRINSIC PRINT,READ,ASCII,BINARY;
00007000 00000 1
00008000 00000 1
00009000 00000 1
00010000 00000 1
00011000 00000 1
00012000 00000 1
00013000 00000 2
00014000 00000 2
00015000 00000 2
00016000 00013 2
00017000 00015 2
00018000 00024 2
00019000 00000 1
00020000 00000 1
00021000 00000 1
00022000 00000 1
00023000 00000 1
00024000 00000 1
00025000 00000 1
00026000 00000 1
00027000 00000 1
00028000 00000 2
00029000 00001 2
00030000 00007 2
00031000 00013 2
00032000 00014 2
00033000 00000 1
00034000 00000 1
00035000 00000 1
00036000 00000 1
00037000 00000 1
00038000 00000 2
00039000 00000 2
00040000 00006 2
00041000 00006 2
00042000 00007 2
00043000 00000 1
00044000 00000 1
00045000 00000 1
00046000 00000 1
00047000 00000 1
00048000 00004 1
00049000 00005 1
00050000 00005 1
PRIMARY DB STORAGE=3012;
NO. ERRORS=000;
PROCESSOR TIME=0:00:05;
SECONDARY DB STORAGE=$00020
NO. WARNINGS=000
ELAPSED TIME=0:01:02

```

<<***** SEGMENT 1 *****>>

<<***** SEGMENT 2 *****>>

<<***** MAIN SEGMENT *****>>

SUBPROGRAM

```

00001000 00000 0 $CONTROL
00002000 00000 0 BEGIN
00003000 00000 1 INTEGER I,J,K;
00004000 00000 1 REAL X,Y,Z;
00005000 00000 1 ARRAY ARY(0:15);
00006000 00000 1 INTRINSIC PRINT,READ,ASCII,BINARY;
00007000 00000 1
00008000 00000 1
00009000 00000 1
00010000 00000 1
00011000 00000 1
00012000 00000 1
00013000 00000 2
00014000 00000 2
00015000 00000 2
00016000 00013 2
00017000 00016 2
00018000 00024 2
00019000 00000 1
00020000 00000 1
00021000 00000 1
00022000 00000 1
00023000 00000 1
00024000 00000 1
00025000 00000 1
00026000 00000 1
00027000 00000 1
00028000 00000 2
00029000 00001 2
00030000 00007 2
00031000 00013 2
00032000 00014 2
00033000 00000 1
00034000 00000 1
00035000 00000 1
00036000 00000 1
00037000 00000 1
00038000 00000 2
00039000 00000 2
00040000 00006 2
00041000 00006 2
00042000 00007 2
00043000 00000 1
00044000 00000 1
00045000 00000 1
00046000 00000 1
00047000 00000 1

```

<<***** SEGMENT 1 *****>>

```

$CONTROL SEGMENT=SEG1
PROCEDURE ONE(A)
BEGIN

```

```

ENTRY ENTRY1;
INTEGER II,J,KK;
II := I+J*K*KK;
ENTRY1;
J := READ(ARY(3),3);
END;

```

<<***** SEGMENT 2 *****>>

```

$CONTROL SEGMENT=SEG2
PROCEDURE TWO(A)
X := Y+Z*REAL(I+J*K);

```

```

PROCEDURE TWO(B)
BEGIN
ONE(A);
X := Y*Y+Z;
I := I+3;
J := K+1
END;

```

```

PROCEDURE TWO(C(P1,P2,P3))
VALUE P1,P3;
INTEGER P1,P2,P3;
BEGIN
ENTRY ENTRY2;
I := J/K+3;
ENTRY2;
X := X/Y
END;

```

<<***** MAIN SEGMENT *****>>

```

$CONTROL SEGMENT=MANE
TWO(C(33,I,J));
PRIMARY CH STORAGE=$012; SECONDARY OR STORAGE=$00020
NO. ERRORS=000; NO. WARNINGS=000
PROCESSOR TIME=0:00:06; ELAPSED TIME=0:00:55

```

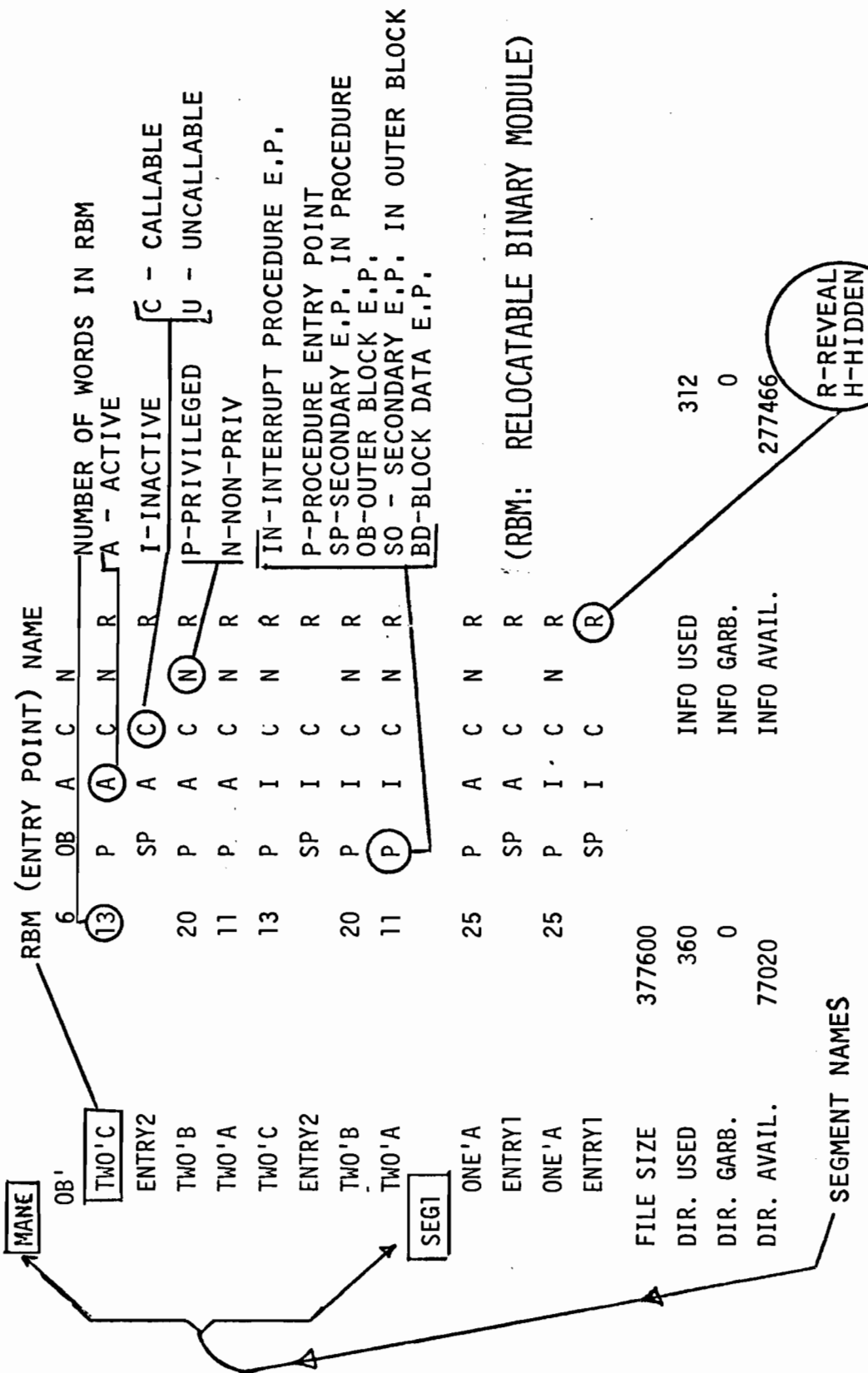
LISTUSL
:SEGMENTER

SEGMENTER SUBSYSTEM (1.0)

-USL SEGUSL

-(LISTUSL)

USL FILE SEGUSL.PUB.SYS



NOTE: SEE FIGURE 7-1 OF MPE MANUAL

<<PRIVILEGED>> <ALLOCATED> <RESIDENT> <SYSTEM>>

BUILD SL SEGSL, 500, 1

ADD SL SEG1

ADD SL SEG2

[LISTSL]

SL FILE SEGSL.PUB.SYS

SEGMENT	0	SEG1	LENGTH	34
* ENTRY POINTS *				
ONE 'A	0			
ENTRY1	0			
* EXTERNALS *				
READ	0			

STT NUMBER	C-CALLUABLE	U-UNCALLABLE
1 C		
2 C		

10

SEGMENT	1	SEG2	LENGTH	54
* ENTRY POINTS *				
TWO 'A	0			
TWO 'B	0			
TWO 'C	0			
ENTRY2	0			
* EXTERNALS *				
ONE 'A	0			
ITOI'	0			

PARAMETER CHECKING

0-NO CHECKING

1-FOR PROCEDURE TYPE ONLY

2-FOR PROCEDURE TYPE AND NUMBER OF PARAM.

3-FOR PROC. TYPE AND NUMBER AND TYPE PARAM.

AVAILABLE 172000

SEGMENT NUMBER OF REFERENCE

REFERENCE TO OTHER SEGMENTS

3000

USED

11

6	0
5	?

:PREP SEGUSL,SEGPF,PMAP

PROGRAM FILE SEGPF.PUB.SYS ; PMAP

MANE	STT NUMBER	SEGMENT OF REFERENCED PROCEDURE
OB'	0	
TWO'C	1	1
ONE'A	2	2
TERMINATE'	3	?
SEGMENT LENGTH	4	
SEG2	14	
TWO'C	1	
ITOI'	0	
ENTRY2	5	?
TWO'B	2	
ONE'A	3	
TWO'A	6	
SEGMENT LENGTH	13	
SEG1	33	
ONE'A	54	
READ	2	
ENTRY1	1	
SEGMENT LENGTH	4	
PRIMARY DB	12	
SECONDARY DB	20	
TOTAL DB	32	
SEGMENT NAME		

CAPABILITY 1773
TOTAL CODE 124
TOTAL RECORDS 7

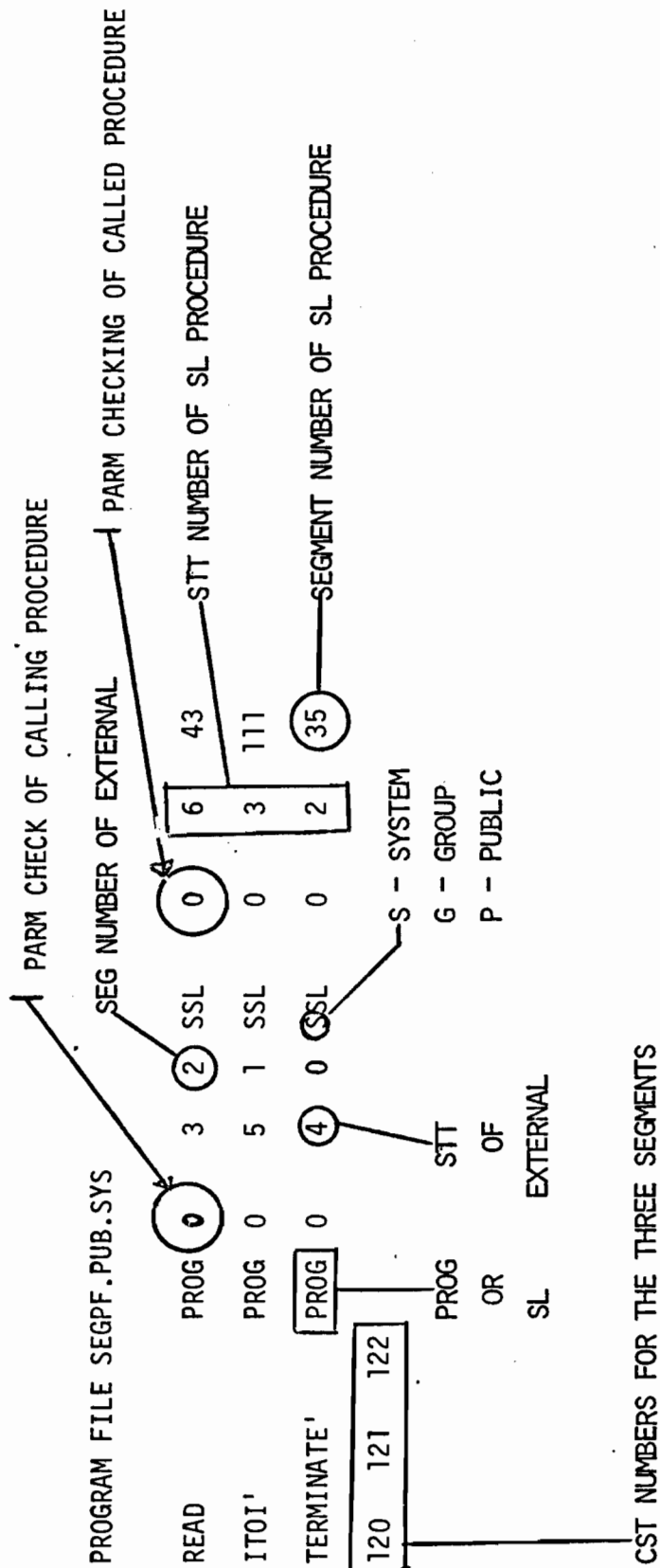
INITIAL STACK 1440
INITIAL DL 0
MAXIMUM DATA ?

STARTING ADR. OF ENTRY POINT
14



LMAP

:RUN SEGPF;LMAP



TEXT EDITOR

ADVANCED FEATURES

stam shell

ADVANCED EDITOR FEATURES

● CALLABLE COMMAND STRINGS

Z: :=
USE

● REPETATIVE COMMAND EXECUTION

WHILE
BEGIN
END
NOT
OR
YES

● SPL/FORTRAN CODE CALLABLE FROM EDITOR

PROCEDURE

Z:: = COMMAND

/TEXT TEXTF

/FIND LAST

61

/Z::=

INITIALLY SET BY Z::=

FIND FIRST

/

Z::=

1

CALLED BY Z::=

WHAT IS A GOOD PROGRAM? *

Z

FIND 28

/

RESET BY Z

Z::=

28

VIRTUES HERE, AS WELL AS GOOD DOCUMENTATION.

RESET TO MULTIPLE COMMANDS.

Z

FIND FIRST;FIND 28;FIND LAST

/

Z::=

1

WHAT IS A GOOD PROGRAM? *

28

VIRTUES HERE, AS WELL AS GOOD DOCUMENTATION.

61

USE COMMAND

Q COMMAND

```
ADD
  1  FINDQ FIRST;
  2  Q"#L1";
  3  //
...
/KEEP USEFILE1
```

CREATE USE FILE

KEEP USE FILE

```
DELETE ALL
/ADD
  1  ADD;
  2  FIND LAST;
  3  USE USEFILE1;
  4  FIND "Q";
  5  //
...
/KEEP USEFILE2
```

CREATE SECOND USE FILE

KEEP SECOND USE FILE

```
/USE USEFILE2
  5  LINE 5
  6  LINE 6
  7  //
...
  6  LINE 6
  †
#L1
  4  FIND "Q";
  †
/
```

CALL SECOND USE FILE

ADD COMMAND

FIND LAST COMMAND

FIRST USE FILE CALLED

FIND "Q" COMMAND

PROGRAMMABLE EDITS

WHILE COMMAND

FIND FIRST

1

WHAT IS A GOOD PROGRAM? *

WHILE

/ FINDQ "PROGRAMMER"

/ LIST *

WHILE INITIALIZES SPECIAL INTERPRETER
SECOND COMMAND STARTS WHILE LOOP

/FINDQ "PROGRAMMER"

/LIST *

7 SUCCESSFUL, AND THIS GOAL IS THE PRIMARY AIM OF A PROGRAMMER.

/FINDQ "PROGRAMMER"

/LIST *

9 BECOME MORE IMPORTANT AS A PROGRAMMER GAINS EXPERIENCE AND AS

/FINDQ "PROGRAMMER"

/LIST *

19 PROGRAMMER MUST USE AS FEW INSTRUCTIONS AS POSSIBLE AND CHOOSE

/FINDQ "PROGRAMMER"

/LIST *

21 CYCLES THAT ARE REQUIRED. ALSO THE PROGRAMMER MUST USE INPUT-

/FINDQ "PROGRAMMER"

/LIST *

24 PROGRAMMER. QUITE OFTEN A PROGRAMMER BECOMES SO ANXIOUS TO SAVE

/FINDQ "PROGRAMMER"

/LIST *

32 DESIGNS A PROGRAM, THE PROGRAMMER SHOULD TRY TO ANTICIPATE AS

/FINDQ "PROGRAMMER"

/LIST *

52 THIS SITUATION HAS OCCURRED MANY TIMES. IT IS UP TO THE PROGRAMMER

/FINDQ "PROGRAMMER"

*21*STRING NOT FOUND BEFORE LIMIT

AT DEPTH 2

-/LIST *

FIND FAILURE KILLS SECOND COMMAND

WHILE - BEGIN - END COMMANDS

TEXT TEXTF

IF IT IS OK TO CLEAR RESPOND "YES"

CLEAR? Y

/WHILE

/ FINDQ "PROGRAMMER"

FIRST WHILE COMMAND

/ BEGIN

BEGIN ALLOWS MORE COMMANDS

/ CHANGE "PROGRAMMER" TO "PGMR"

/ LIST * OFFLINE

/ END

END STARTS WHILE LOOP

↑
2 line/page
on LP.

/FINDQ "PROGRAMMER"

/BEGIN

/CHANGE "PROGRAMMER" TO "PGMR"

7 SUCCESSFUL, AND THIS COAL IS THE PRIMARY AIM OF A PGMR.

/LIST * OFFLINE

/END

/FINDQ "PROGRAMMER"

/BEGIN

/CHANGE "PROGRAMMER" TO "PGMR"

19 PGMR MUST USE AS FEW INSTRUCTIONS AS POSSIBLE AND CHOOSE

/LIST * OFFLINE

/END

/FINDQ "PROGRAMMER"

/BEGIN

/CHANGE "PROGRAMMER" TO "PGMR"

24 PGMR. QUITE OFTEN A PROGRAMMER BECOMES SO ANXIOUS TO SAVE

/LIST * OFFLINE

/END

/FINDQ "PROGRAMMER"

/BEGIN

/CHANGE "PROGRAMMER" TO "PGMR"

32 DESIGNS A PROGRAM, THE PGMR SHOULD TRY TO ANTICIPATE AS

/LIST * OFFLINE

/END

/FINDQ "PROGRAMMER"

/BEGIN

/CHANGE "PROGRAMMER" TO "PGMR"

52 THIS SITUATION HAS OCCURRED MANY TIMES. IT IS UP TO THE PGMR,

/LIST * OFFLINE

/END

/FINDQ "PROGRAMMER"

*21*STRING NOT FOUND BEFORE LIMIT

AT DEPTH 2

-/BEGIN

-/CHANGE "PROGRAMMER" TO "PGMR"

-/LIST * OFFLINE

-/END

FIND FAILURE SETS

FLAG TO FALSE-KILLS

REST OF COMMAND

TEXT TEXTF

IF IT IS OK TO CLEAR RESPOND "YES"

CLEAR? Y/LIST 10

10 PROGRAMMING PROBLEMS BECOME MORE INVOLVED.

/

FIND FIRST

1

/

WHILE/ FIND "PROGRAMMING"(+10)/ BEGIN/ FIND "PROBLEMS"/*(+10)/ LIST */ END*phantom* WHAT IS A GOOD PROGRAM? **wh*
*= */*(+10) → look current pos + 10*

(+10) SETS POINTER TO END OF STRING

/*(+10) SEARCHES NEXT 10 CHARACTERS FOR
SECOND STRING

/FIND "PROGRAMMING"(+10)

10 PROGRAMMING PROBLEMS BECOME MORE INVOLVED. FIRST STRING FOUND

/BEGIN

/FIND "PROBLEMS"/*(+10)

SECOND STRING FOUND

10 PROGRAMMING PROBLEMS BECOME MORE INVOLVED.

/LIST *

LIST COMMAND EXECUTES

10 PROGRAMMING PROBLEMS BECOME MORE INVOLVED.

/END

/FIND "PROGRAMMING"(+10)

FIRST STRING FOUND

49 *WALNUT, FRANCIS K., INTRODUCTION TO COMPUTER PROGRAMMING AND

/BEGIN

/FIND "PROBLEMS"/*(+10)

*21*STRING NOT FOUND BEFORE LIMIT
AT DEPTH 3

SECOND STRING NOT FOUND

-/LIST *

REST OF COMMANDS FLUSHED

-/END

/FIND "PROGRAMMING"(+10)

*21*STRING NOT FOUND BEFORE LIMIT
AT DEPTH 2

-/BEGIN

-/FIND "PROBLEMS"/*(+10)

-/LIST *

-/END

/

NOT COMMAND

*REVERSES FLAG SETTING OF NEXT
EXPRESSION*

```
FINDQ FIRST
/WHILE
/  FIND "PROGRAMMING"(+10)
/  BEGIN
/    NOT
/    FINDQ "PROBLEMS"/*(+10)
/    LIST *
/    END
```

```
/FIND "PROGRAMMING"(+10)    STRING FOUND - FLAG = TRUE
10    PROGRAMMING PROBLEMS BECOME MORE INVOLVED.
```

```
/BEGIN
/NOT
/FINDQ "PROBLEMS"/*(+10)    SECOND STRING FOUND-FLAG=TRUE
                              HOWEVER NOT CHANGES FLAG=FALSE
-/LIST * ← REST OF COMMANDS FLUSH
-/END
```

```
/FIND "PROGRAMMING"(+10)
49    *WALNUT, FRANCIS K., INTRODUCTION TO COMPUTER PROGRAMMING AND
```

```
/BEGIN
/NOT
/FINDQ "PROBLEMS"/*(+10)    SECOND STRING NOT FOUND-FLAG=FALSE
*21*STRING NOT FOUND BEFORE LIMIT NOT CHANGES FLAG=TRUE
AT DEPTH 4
/LIST *    REST OF COMMANDS EXECUTE
49    *WALNUT, FRANCIS K., INTRODUCTION TO COMPUTER PROGRAMMING AND
/END
/FIND "PROGRAMMING"(+10)
*21*STRING NOT FOUND BEFORE LIMIT
AT DEPTH 2
-/BEGIN
-/NOT
-/FINDQ "PROBLEMS"/*(+10)
-/LIST *
-/END
```

OR COMMAND

*SKIPS NEXT COMMAND FLAG=TRUE
SETS FALSE FLAG = TRUE

```
FINDQ FIRST  
/WHILE  
/ FIND "PROGRAMMING"(+10)  
/ BEGIN  
/ FIND "PROBLEMS"/*(+10)  
/ OR  
/ LISTQ *  
/ END
```



```
/FIND "PROGRAMMING"(+10)  
10 PROGRAMMING PROBLEMS BECOME MORE INVOLVED.  
↑
```

```
/BEGIN  
/FIND "PROBLEMS"/*(+10) STRING FOUND  
10 PROGRAMMING PROBLEMS BECOME MORE INVOLVED.  
↑
```

```
/OR FLAG = TRUE - LIST COMMAND SKIPPED
```

```
/LISTQ *  
/END
```

```
/FIND "PROGRAMMING"(+10)  
49 *WALNUT, FRANCIS K., INTRODUCTION TO COMPUTER PROGRAMMING AND  
↑
```

```
/BEGIN  
/FIND "PROBLEMS"/*(+10)  
*21*STRING NOT FOUND BEFORE LIMIT  
AT DEPTH 3
```

```
-/OR FLAG = FALSE - RESET TO = TRUE  
-/LISTQ * LIST COMMAND EXECUTES  
*WALNUT, FRANCIS K., INTRODUCTION TO COMPUTER PROGRAMMING AND CODING,  
/END
```

```
/FIND "PROGRAMMING"(+10)  
*21*STRING NOT FOUND BEFORE LIMIT  
AT DEPTH 2
```

```
-/BEGIN  
-/FIND "PROBLEMS"/*(+10)  
-/OR  
-/LISTQ *  
-/END  
/
```

YES COMMAND

*SETS FALSE FLAG = TRUE

FIND FIRST

WHAT IS A GOOD PROGRAM? *

```

/WHILE
/  FIND "PROGRAMMING"(+10)
/  BEGIN
/    FIND "PROBLEMS"/*(+10)
/    CHANGE "PROBLEMS" TO "PROB." IN *
/    YES
/    LIST *
/    END
/FIND "PROGRAMMING"(+10)
10    PROGRAMMING PROBLEMS BECOME MORE INVOLVED.
      ↑
/BEGIN
/FIND "PROBLEMS"/*(+10)  STRING FOUND - FLAG = TRUE
10    PROGRAMMING PROBLEMS BECOME MORE INVOLVED.
      ↑
/CHANGE "PROBLEMS" TO "PROB." IN *
10    PROGRAMMING PROB. BECOME MORE INVOLVED.
/YES    YES-NO ACTION ON FLAG = TRUE
/LIST *
10    PROGRAMMING PROB. BECOME MORE INVOLVED.
/END
/FIND "PROGRAMMING"(+10)
49    *WALNUT, FRANCIS K., INTRODUCTION TO COMPUTER PROGRAMMING AND
      ↑
/BEGIN
/FIND "PROBLEMS"/*(+10)
*21*STRING NOT FOUND BEFORE LIMIT  FLAG = FALSE
AT DEPTH 3
-/CHANGE "PROBLEMS" TO "PROB." IN *  CHANGE COMMAND FLUSHED
-/YES    YES - CHANGES FLAG = TRUE
/LIST *  LIST EXECUTES
49    *WALNUT, FRANCIS K., INTRODUCTION TO COMPUTER PROGRAMMING AND
/END
/FIND "PROGRAMMING"(+10)
*21*STRING NOT FOUND BEFORE LIMIT
AT DEPTH 2
-/BEGIN
-/FIND "PROBLEMS"/*(+10)
-/CHANGE "PROBLEMS" TO "PROB." IN *
-/YES
-/LIST *
-/END
/

```

CONTROLLING THE WHILE COMMAND

SET TIME=7 SET TIME LIMITS WHILE ITERATIONS

WHILE;FINDQ FIRST;LIST 3 OFFLINE

/FINDQ FIRST;
/LIST 3 OFFLINE
/FINDQ FIRST;
/LIST 3 OFFLINE
/FINDQ FIRST;
/LIST 3 OFFLINE
/FINDQ FIRST;
/LIST 3 OFFLINE
/FINDQ FIRST;
/LIST 3 OFFLINE
/FINDQ FIRST;
/LIST 3 OFFLINE
/FINDQ FIRST;
/LIST 3 OFFLINE

53

/

/SET TIME=3

/SET SHORT

/WHILE

/ FINDQ FIRST

/ LIST ALL OFFLINE

53

TIME-OUT ON WHILE ITERATION

/

SET SHORT INHIBITS VERBALIZATION

FORTRAN SUBROUTINE

XXXEDIT

BUILD XXXUSL;CODE=1024

BUILD USL FILE

FORTRAN EDPROC,XXXUSL

COMPILE SOURCE FILE EDPROC

PAGE 0001 HEWLETT-PACKARD 32102A.00.3 FORTRAN/3000 WED, MAY 9,
1973, 3:25 PM

```

$CONTROL USLINIT,SEGMENT=XXXSEG    SEGMENT NAME USED
      LOGICAL FUNCTION XXXEDIT (STRING,SIZE,NUMBER,SPACE)
      INTEGER SIZE,SPACE(20),IC     NAME OF SUBROUTINE
      EQUIVALENCE (IC,C)
      CHARACTER*80 STRING,NUMBER*8,SC*10,C*2
      IC=0
      SC='%#@&.,: "$/'
      DO 10 I=1,SIZE
      IF (STRING[I:1].LT.'0'.OR.STRING[I:1].GT.'9')GOTO10
      C[I:1]=STRING[I:1]
      IC=IC+47
      STRING[I:1]=SC[IC:1]
10    CONTINUE
      XXXEDIT=.TRUE.  =.TRUE. MUST END SUBROUTINE IF SUCCESSFUL
      RETURN
      END
  
```

**** NO ERRORS, NO WARNINGS; PROGRAM UNIT COMPILED ****
COMPILATION TIME 0.972 SECONDS

END OF PROGRAM

SUBROUTINE FUNCTION:

CONVERT

1 TO #	6 TO :
2 TO @	7 TO "
3 TO &	8 TO \$
4 TO .	9 TO /
5 TO ,	0 TO %

PROCEDURE COMMAND
PREPARATION AND USE

:SEGMENTER

CALL SEGMENTER

SEGMENTER SUBSYSTEM (1.0)

USL XXXUSL

IDENTIFY USL FILE TO BE USED

BUILDSL SL,500,1

CREATE SEGMENT LIBRARY FILE (DO NOT ISSUE
IF ALREADY EXISTS!)

-SL SL

IDENTIFY SEGMENT LIBRARY TO BE USED AS TARGET
OF ADDSL COMMAND

-ADDSL XXXSEG

PREP XXXSEG AND PLACE IN SL

-EXIT

LEAVE SEGMENTER

END OF PROGRAM
:

*MUST BE IN
AN SL.*

EDITOR

HP32201A.01.4 EDIT/3000 WED, MAY 9, 1973, 3:38 PM

ADD

1 A1B2C3D4E5F6G7H8I9J0
2 1234567890
3 ABCDEFGHIJ
4 0987654321
5 //

CREATE TEXT FILE

...

/PROCEDURE XXXEDIT,G,ALL

TEXT FILE RANGE

PROCEDURE COMMAND LOADS SUBROUTINE
XXXEDIT FROM GROUP SL AND EXECUTES IT
AGAINST THE RANGE "ALL"

/LIST ALL

1 A#B0C&D.E,F:G"HSI/JZ
2 #0&.,:"S/%
3 ABCDEFGHIJ
4 %/\$":.,.&0#