fig-FORTH on a PDP-11 Hard Disk

Paul Hardy

18 November 2017

“Please, sir, I want some more.”

—Oliver Twist

Raw Disk Space

RX01: *250 kbytes*

RL02: *10,240 kbytes*

RT-11 Disk Structure

Disk Byte	Usage
$1{,}000_8$	Home Block
$6{,}000_8$	RT-11 Directory
$16{,}000_8$	`FORTH.DAT`
$40{,}016{,}000_8$	`FORTH.MAC`, etc.
$47{,}754{,}000_8$	Bad Sector Map
$47{,}777{,}777_8$	Last Disk Byte

`FORTH.DAT`: **8192 Screens**

RL02 Disk Geometry

15 … 7	6	5 … 0
CYLINDER	HD	SECTOR

- **512** cylinders / disk
- **2** heads / cylinder
- **40** sectors / track
- **256** bytes / sector

10,240 kbytes

RL02 Disk Geometry

15 … 7	6	5 … 0
CYLINDER	HD	SECTOR

- **512** cylinders / disk
- **2** heads / cylinder
- **40** sectors / track
- **256** bytes / sector

10,240 kbytes

Exactly 10 Forth Screens per Track

FORTH.MAC RL02 Option

```
;
;RT11=1   ; COMMENTED OUT UNLESS RT-11
;RSX11=1  ; COMMENTED OUT UNLESS RSX11M
ALONE=1   ; COMMENTED OUT UNLESS STAND-ALONE
RL02=1    ; COMMENTED OUT UNLESS STAND-ALONE RL02 IMAGE
          ;       (ALSO UN-COMMENT ALONE=1 FLAG)
EIS=1     ; COMMENTED OUT UNLESS HARDWARE MULTIPLY-DIVIDE
;LINKS=1  ; COMMENTED OUT UNLESS SUBROUTINE LINKAGE FROM
;                 FORTH TO OTHER LANGUAGES
;
```

1. RL02 Trivial Boot

```
sh-3.2$ pdp11

PDP-11 simulator V4.0-0 Beta        git commit id: a719ef51
sim> attach rl0 rl02-boot.dsk
sim> boot rl
>
HALT instruction, PC: 000224 (HALT)
sim> quit
Goodbye
sh-3.2$
```

2. Forth Binary Boot

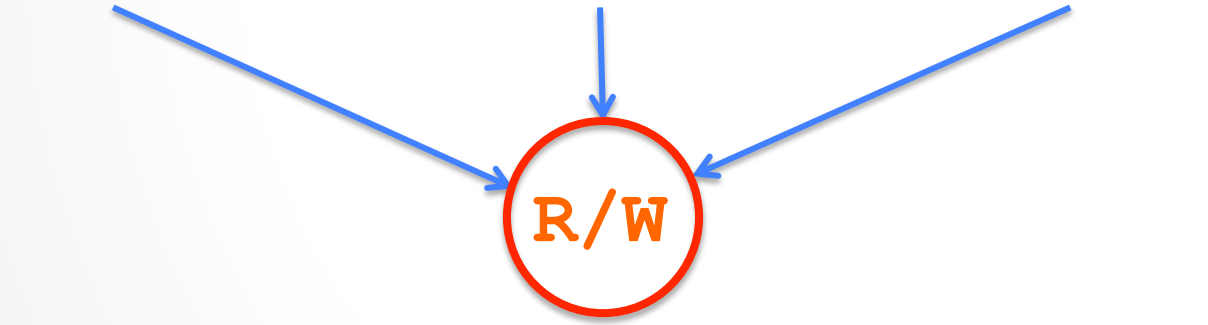
```
sh-3.2$ pdp11

PDP-11 simulator V4.0-0 Beta        git commit id: a719ef51
sim> attach rl0 rl02-forth.dsk
sim> boot rl

FIG-FORTH  V 1.3.2
1 2 + . 3  OK
bye
HALT instruction, PC: 015270 (BIC @52122(R2),R4)
sim> q
Goodbye
sh-3.2$
```

3. Add RL02 Support: R/W

```
R/W ( ADDR SCREEN# FLAG:R=1,W=0 → )
```



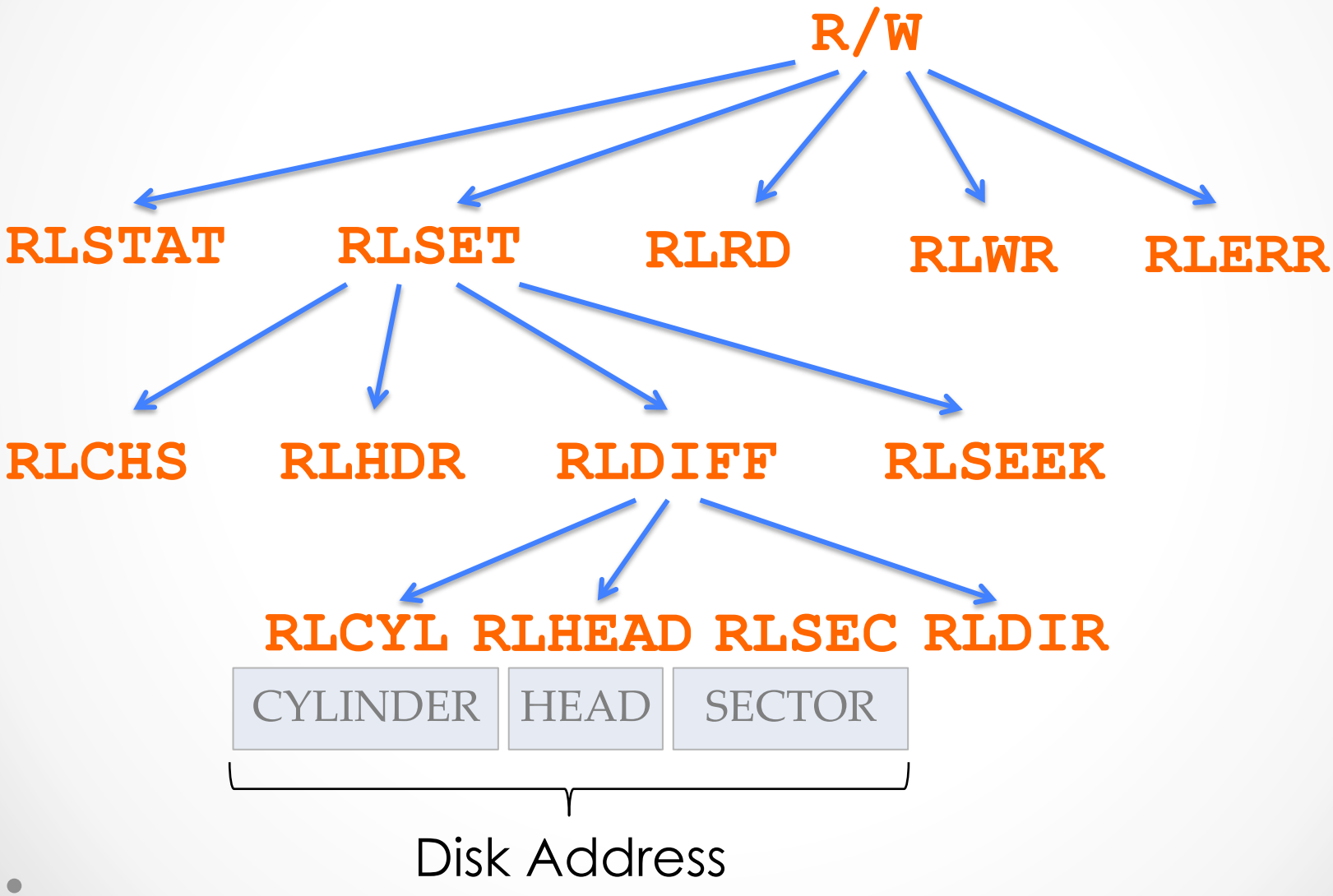
Already in FORTH.MAC

Need RL02 Version

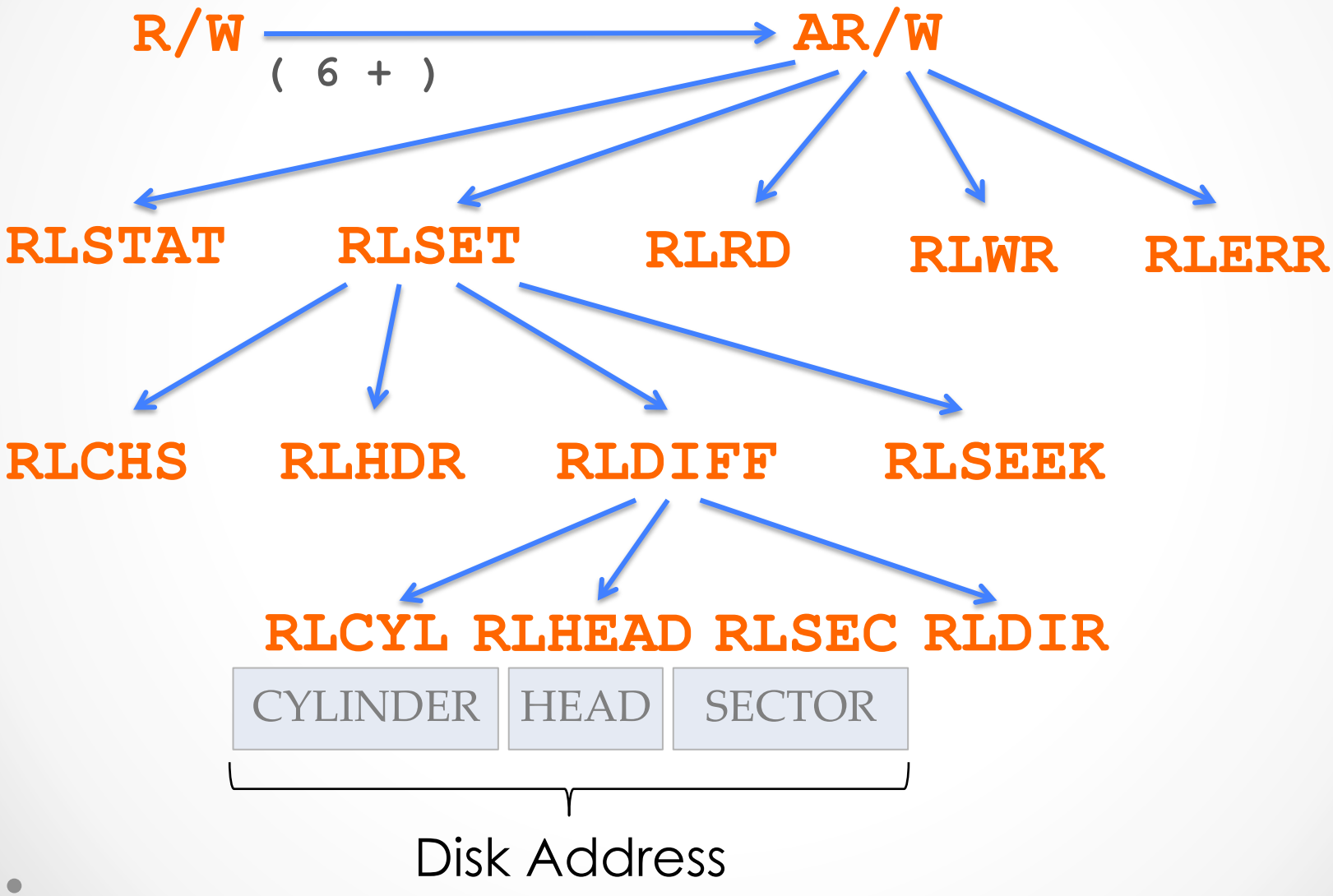
R/W and the 3 'R's: Reading & Writing & 'Rithmetic

1. Swap & add 6 to **SCREEN#** for absolute **BLOCK#** on RL02
2. Seek: **RLSET** (**BLOCK#** → **CYL|HD|SEC SEEKSTATUS**)
   a. **RLCHS**: Calculate **Cylinder|Head|Sector** from **BLOCK#**
   b. **RLHDR**: Get next disk **sector header**: **Read Header**
   c. **RLDIFF**: Calculate **relative offset** to desired position
   d. **RLSEEK**: **Seek** by relative offset from current position
   e. Return **CYL|HD|SEC SEEKSTATUS**
3. Read or Write 1024 Bytes to/from Memory:
   - **RLRD** (**ADDR CYL|HD|SEC** → **STATUS:0=NO-ERROR**)
   - **RLWR** (**ADDR CYL|HD|SEC** → **STATUS:0=NO-ERROR**)

RL02 Forth Words



RL02 Forth Words



3. Forth RL02 Support

```
sh-3.2$ pdp11

PDP-11 simulator V4.0-0 Beta        git commit id: a719ef51
sim> att rl0 rl02_figforth-1.3.3.dsk
sim> b rl

FIG-FORTH  V 1.3.3
1 load
LOADING EDITOR... R ISN'T UNIQUE I ISN'T UNIQUE
LOADING ASSEMBLER... R0 ISN'T UNIQUE # ISN'T UNIQUE
LOADING STRING PACKAGE...
LOADING STRING EXTENSIONS...
BYE ISN'T UNIQUE
 OK
```

☺ !!!

New PDP-11 **DUMP** Word

```
octal  OK
1050 40 dump
 ADDR    VALUE      BYTES      RAD50  ASCII
001050 000000  (   0 |    0)  /    /  '  '
001052 046203  (114 | 203)  /LI$/  ' L'
001054 152111  (324 | 111)  /38Y/  'IT'
001056 000000  (   0 |    0)  /    /  '  '
001060 001062  (   2 |   62)  / NB/  '2 '
001062 012445  ( 25 |   45)  /COM/  '% '
001064 012402  ( 25 |    2)  /CNR/  '  '
001066 000132  (   0 | 132)  / BJ/  'Z '
001070 042607  (105 | 207)  /KD9/  ' E'
001072 042530  (105 | 130)  /KC2/  'XE'
001074 052503  (125 | 103)  /MY$/  'CU'
001076 142524  (305 | 124)  /1V6/  'TE'
001100 001052  (   2 |   52)  / M4/  '* '
001102 001104  (   2 | 104)  / NT/  'D '
001104 012502  ( 25 | 102)  /CPB/  'B '
001106 000132  (   0 | 132)  / BJ/  'Z '
 OK
```

LIT

MOV (IP)+,-(S)

NEXT

EXECUTE

MOV (S)+, W
JMP @(W)+

Words for *Starting Forth*

```
[79 83 index

 79 ( DEFINITIONS FOR "STARTING FORTH" - NUMBERS )
 80 ( DEFINITIONS FOR "STARTING FORTH" - STACK )
 81 ( DEFINITIONS FOR "STARTING FORTH" - STACK, CONT'D )
 82 ( DEFINITIONS FOR "STARTING FORTH" - I/O, ETC. )
 83 ( MISCELLANEOUS DEFINITIONS FOR "STARTING FORTH" ) OK
  OK
[79 load
TYPE 'FORGET STARTING' TO BACK OUT THESE DEFINITIONS
 OK
```

Words for *Starting Forth*

Written by Robert L. Smith

```
vlist
DU<   DMIN   DMAX   D=   D0=   2VARIABLE   2CONSTANT   U/MOD   EXIT
2!   2@   U.R   PAGE   VTCLEOS   VTHOME   VTCLSCROLL   >IN   H
BLANK   K'   K   J'   J   I'   'R   TUCK   NIP   R@   2ROT   2OVER
ROLL   PICK   2SWAP   2DROP   2DUP   ?DUP   -ROT   .S   DEPTH   CLEAR
'S   INVERT   M+   DNEGATE   D-   NEGATE   2/   ASR   2*   0.   STARTING
```

Written by Paul Hardy

Credits

```
2 list
SCR # 2
  0 ( MISCELLANEOUS FIG-FORTH DEFINITIONS )
  1 : MOVE MOVEW ;
  2 : DLIST VLIST ;
  3
  4
  5
  6
  7
  8 ( FORTH.DAT AUTHORS: )
  9 (    SCREENS 6, 7, 8, 9: BILL RAGSDALE, 1979 )
 10 (    SCREENS 18, 23, 57, 58, 79-82: PAUL HARDY, 2017 )
 11 (    SCREEN 83: ROBERT L. SMITH, 1981 )
 12 (    ALL OTHER SCREENS: JOHN S. JAMES, 1979-1980 )
 13
 14
 15
 OK
```

Possible Future Work

- Support 4 RL02 Drives on One Disk Controller
- Support KW11-L and/or KW11-P Clock
- RT-11 File System Year Rollover

PDP-11s ❤ FIG Forth

- Shared 16-bit Architecture
- Native Standalone Forth
- Powerful Macro Assembler
- `NEXT` is only 2 Instructions

PDP-11s Forever!

The pdp11 is dead.

Long live the > pdp11 !

Simulators keep the legend alive.

Resources

- Bootable FIG Forth Disk Images, source files, utilities, etc.:
  http://www.stackosaurus.com/figforth
- RT-11v4 & v5.3 (note hobbyist license):
  http://simh.trailing-edge.com/software.html
- Ersatz-11 (Demo Version):
  http://www.dbit.com/demo.html
- PUTR: http://www.dbit.com/putr/
- Empty PDP-11 Disk Images (for system generation):
  http://www.dbit.com/pub/pdp11/empty/
- SIMH: http://simh.trailing-edge.com/
- Original Forth Interest Group Files:
  http://www.forth.org/fig-forth/contents.html

Backup

RL02 Strategy

1. Boot Sector: Output "**>**" to Terminal

Verify boot block placement on disk

1. Trivial Boot: Output '>'

```
;
; Declarations for terminal I/O
;
RCSR=177560      ; Terminal receive control and status
RBUF=177562      ; Terminal receive buffer
XCSR=177564      ; Terminal transmit control and status
XBUF=177566      ; Terminal transmit buffer

        .ASECT
.=0
BOOTRL:
TXWAIT: TST      @#XCSR          ; Wait until terminal is ready to transmit
        BEQ      TXWAIT
        MOV      #76,@#XBUF      ; Output a '>' on terminal
TXOUT:  TST      @#XCSR          ; Wait until terminal displays character
        BEQ      TXOUT
        HALT
        .END
```

RL02 Strategy

1. Boot Sector: Output "**>**" to Terminal

   Verify boot block placement on disk

2. Load Screens 40.25 – 47; Run Forth

   Not yet able to load FORTH.DAT

2. Loading **FORTH.MAC**

- $134{,}000_8$: **FORTH.DAT** Screen 40 Start
- **$134{,}400_8$: Start Loading at 000400_8**
- **$135{,}000_8$: Forth Binary Beginning**
- **$153{,}777_8$: Stop Loading**
- $154{,}000_8$: **FORTH.DAT** Screen 48 Start

RL02 Strategy

1. Boot Sector: Output "**>**" to Terminal

   Verify boot block placement on disk

2. Load Screens 40.25 – 47; Run Forth

   Not yet able to load FORTH.DAT

3. Add RL02 Support to **`FORTH.MAC`**

   Able to load FORTH.DAT – everything works ☺

Modifying FORTH.MAC

- Forth Words using Macro-11 Definitions
- Forth Words using PDP-11 Op Codes

Macro-11 Forth Definitions 1

Forth Name

Macro-11 Name

Comment to Line End

```
HEAD   203,R/W,327,RW,DOCOL                    ; ***** R/W
```

Forth Name Length + 200_8

Last Character ('W') + 200_8 *or* 240_8 if Even Length

DOCOL if Forth macros used; end definition with **.WORD SEMIS**

Macro-11 Forth Definitions 2

```
;  ( READ OR WRITE FORTH SCREEN )
;  : R/W  ( ADDR SCREEN# FLAG → )
;          SWAP 6 + SWAP AR/W ;
;
          HEAD     203,R/W,327,RW,DOCOL     ; ***** R/W
;   READ OR WRITE 1024-BYTE SCREEN.
;   ADDR SCREEN# FLAG(R=1,W=0) ->
          ; ADD 6 TO SCREEN# FOR ABSOLUTE DISK BLOCK#
          .WORD    SWAP,LIT,6,PLUS,SWAP
          .WORD    ARW        ; USE ABSOLUTE DISK BLOCK#
          .WORD    SEMIS
```

Forth word in Macro-11 "Forth" ends with **SEMIS**

Assembler Forth Definitions

Even name length, so use 240_8

```
RLCS=174400

        HEAD    206,RLSTAT,240,RLSTAT       ; ***** RLSTAT
;   GET RL02 DISK STATUS
;   -> RLSTATUS
        MOV     #RLCS,R0
        MOV     #13,4(R0)   ; DISK SETUP FOR GET STATUS:
                            ;   CLEAR ERR REG & GET STATUS
        MOV     #4,(R0)     ; LOAD GET STATUS FUNCTION CODE
        TSTB    (R0)        ; TEST RLCS FOR READY STATE
        BPL     .-2         ; NOT READY YET--KEEP CHECKING
        MOV     6(R0),-(S)  ; PUSH DISK STATUS ONTO STACK
        NEXT
```

No ending **,DOCOL** for Assembler

Forth word in Assembler ends with **NEXT**

Screen 18: PDP-11 Strings

```
18 list
SCR # 18
  0 ( STRING EXTENSIONS FOR PDP-11 )        DECIMAL
  1 CODE >< S () SWAB, NEXT, C;  ( SWAP BYTES )
  2 : LSB 255 AND ;  : MSB >< LSB ;  ( GET LOWER & UPPER BYTES )
  3 : ASCIIRAD ( ASCII -- RAD50 ) DUP 64 > IF 95 AND 64 - ( LETTER )
  4    ELSE DUP 47 > IF 18 - ( DIGIT ) ELSE DUP 32 = IF 0 ELSE DUP
  5   36 = IF 27 ELSE 46 = IF 28 ELSE 29 THEN THEN THEN THEN THEN ;
  6 : RADASCII ( RAD50 -- ASCII ) DUP 0= IF DROP 32 ELSE DUP 27 <
  7    IF 64 + ( LETTER ) ELSE DUP 29 > IF 18 + ( # ) ELSE DUP 27 =
  8   IF DROP 36 ELSE 28 = IF 46 ELSE 42 THEN THEN THEN THEN THEN ;
  9 : RADIX50 ( RAD50 RAD50 RAD50 -- 3RADIX50 ) SWAP ROT ( REVERSE )
 10    40 U* ROT 0 D+ DROP 40 U* ROT 0 D+ DROP ; ( 3 INTO 1 CELL )
 11 : RAD50. DUP 64000 U<  ( PRINT IF WITHIN RADIX-50 RANGE )
 12    IF 0 40 U/ 40 /MOD RADASCII EMIT RADASCII EMIT RADASCII EMIT
 13    ELSE DROP ." ???" THEN ;  ( IF NOT IN RANGE, PRINT "???" )
 14 : ASCII. ( EMIT PRINTABLE OR SPACE ) 127 AND ( IGNORE 8TH BIT )
 15    DUP DUP 31 > SWAP 127 < AND IF EMIT ELSE DROP ."  " THEN ;
 OK
```

Screen 23: PDP-11 Dump

```
23 list
SCR # 23
  0 ( DUMP DEFINITIONS; NEEDS SCREEN 18 )
  1 : HIGHADDR ( ADDR N -- ADDR LIMIT )  ( GET NTH ADDRESS )
  2    OVER 0 ROT 0 D+ 2 0 DMINUS D+ DROP ;
  3 : LOWADDR ( ADDR -- ADDR ) -2 AND ;  ( FORCE LOW BIT TO 0 )
  4 : ADDR+ ( ADDR -- ADDR2 ) 0 2 0 D+ DROP ;  ( UNSIGNED ADD 2 )
  5 : ULIMIT ( ADDRN ADDR1 -- ADDRN ADDR1 0/1 ) OVER OVER U< ;
  6 : 6Z# 0 <# # # # # # # #> TYPE SPACE ; ( PRINT LEADING ZEROES )
  7 : DUMPCELL ( ADDR -- )  ( DUMP 16 BIT CELL ON ONE LINE )
  8    DUP 6Z# @ DUP 6Z#                        ( ADDRESS, CONTENTS )
  9    DUP ."  (" MSB 3 .R ."  | " DUP LSB 3 .R ." ) "  ( 2 BYTES )
 10    DUP ."  /" RAD50. ." /"                          ( RADIX-50 )
 11    DUP ."   '" LSB ASCII. MSB ASCII. ." '" CR ;        ( ASCII )
 12 ( DUMP N BYTES FROM ADDR, ROUNDED TO 16-BIT BOUNDARIES )
 13 : DUMP ( ADDR N -- ) HIGHADDR SWAP LOWADDR
 14    CR ."  ADDR   VALUE     BYTES     RAD50  ASCII" CR
 15    BEGIN DUP DUMPCELL ADDR+ ULIMIT UNTIL DROP DROP ;
 OK
```