What The Heck Was UCSD Pascal?

Richard Kaufmann

THE UCSD P-SYSTEM MUSEUM

IN THE WEST WING OF THE JEFFERSON COMPUTER MUSEUM

TERAK MUSEUM - UCSD PASCAL MUSEUM - USUS LIBRARY
ANCIENT ALPHABETIC ART - LIBRARY - ALTAIR AND IMSAI EMULATORS
REVIVING CASSETTE DATA - DISK UTILITIES - COMPUTER RESCUE
WHAT'S WRONG WITH THIS PICTURE

What is the UCSD P-System?
It is a portable operating system that was popular in the early days of personal computers, in the late 1970s and early 1980s.

Like today's Java, it was based on a "virtual machine" with a standard set of low-level, machine-language-like "p-code" instructions that were emulated on different hardware, including the 6502, the 8080, the Z-80, and the PDP-11. In this way, a Pascal compiler that emitted p-code executables could produce a program that could be run under the P-System on an Apple II, a Xerox 820, or a DEC PDP-11.

The most popular language for the P-System was UCSD Pascal. In fact, the P-System operating system itself was written in UCSD Pascal, making the entire operating system relatively easy to port between platforms.

By writing a p-code interpreter in the platform's native assembly language, and a few minimal hooks to operating system functions for the file system and interacting with the user, you could move a p-code executable from another system and run it on the new platform. In this way, the p-code generated on one computer could be used to bootstrap the port of the P-System to another computer.

Wirth, Jensen and Pascal
Niklaus Wirth first developed the Pascal language around 1969, with the first version implemented on the CDC 6000 in 1970. By 1983, it was an ISO standard language.


Prof. Nicklaus Wirth

http://www.threedee.com/jcm/psystem/index.html

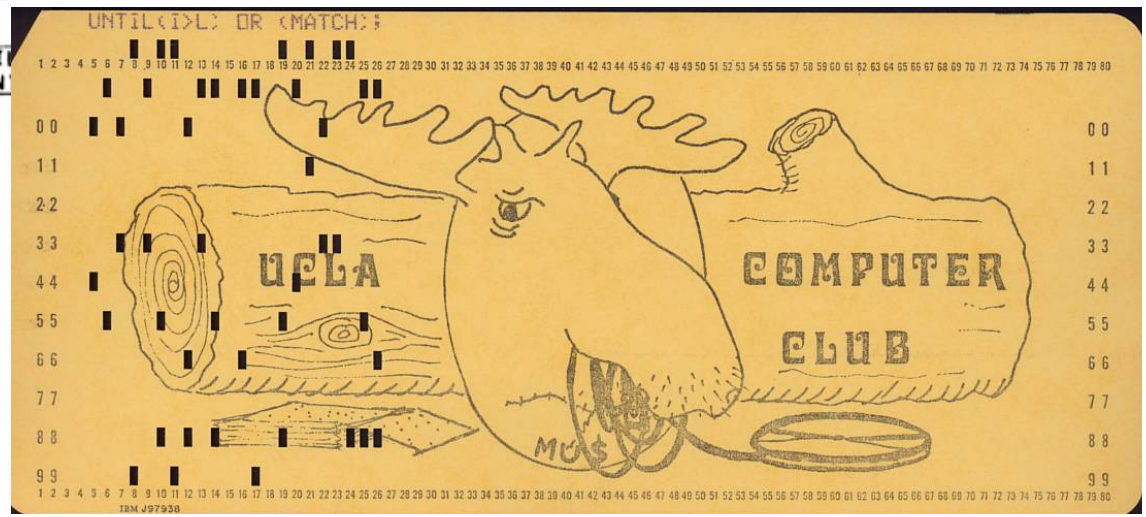
UCSD

Think back to 1974…

- UCSD's main computing environment
  - A Burroughs B6700 "Beast"
    - Algol was the main system language
      - Some amazing architectural features
        - » Stack architecture, "thunks," compiler-enforced security, 48-bit words
      - Kind of an oddity on your résumé, even then…
    - Student access was via punch cards and listings
      - Turnaround times of hours or even days
      - The elite got access to timesharing terminals
    - Usage carefully monitored and billed
      - There are some major cyber-criminals in this room!
  - Some renegade departments with PDP-11s
    - Running Unix v6, RT-11, RSX-11

http://www.inf.ufrgs.br/site/hist/images/b6700_02.jpg

```
                        ELAPSED TIME: 00:00:13

19:13:08          8341  (IU3)PASCALPRR/0008341 REMOVED ON PACK PK066.
19:13:10    EOT   8341  SYSTEM/PASCAL.
                        1.436 SEC CPU, 2.521 SECS IO
                        MEM INTEGRAL: CODE=2.393, DATA=12.360
                        AVERAGE CORE USAGE: CODE=605 DATA=3124
                        ELAPSED TIME: 00:00:23

19:13:10    BOT   8344  (IU3)COMBINATIONS ON PACK.
19:13:11    EOT   8344  (IU3)COMBINATIONS ON PACK.
                        0.546 SEC CPU, 0.667 SECS IO
                        1 CARDS READ  19 LINES PRINTED
                        MEM INTEGRAL: CODE=0.243, DATA=9.152  D/C IN SUBSPACE
                        AVERAGE CORE USAGE: CODE=201 DATA=7546
                        ELAPSED TIME: 00:00:01

19:13:12    EOJ   8332  RICHARD/KAUFMANN.
                        0.256 SEC CPU, 0.614 SECS IO
                        MEM INTEGRAL: CODE=0.077, DATA=1.345
                        AVERAGE CORE USAGE: CODE=88 DATA=1545
                        ELAPSED TIME: 00:00:33


                                                    J O B   C H A R G E S

PROCESSOR SEC          5.8       $.19     37 CARDS          $.02
I/O CHAN. SEC          7.4       $.15     59 LINES          $.06
CORE  KWD-SEC        149.3       $.16
   TOTAL JOB CHARGES             $.58        ACCOUNT BALANCE          $7.65


    UCSD NEWS 10/29/75 11:13      2 LINES
    EARLY CLOSING - THE COMPUTER CENTER WI
     FRIDAY, OCTOBER 31, TO ALLOW THE RUNN
```



Kids these days have it too easy!

Moore's Law: 32 years → $2^{\frac{32}{1.5}}$ = 2,600,000X

- In 1972, a PDP-11/10 consumed half of a 19" rack & ~$30K ($100K adjusted for inflation)
  - One processor
  - 56K of physical memory
    - 16 bit VAs
  - 10MB hard drives were huge and noisy (disks were interchangeable)
  - Dual floppy disk drives (capacity: 160KB each!)
  - Instructions took ~4usec ea.

- An x86 box consumes 1/40th of a 19"rack & ~$10K
  - Two processors (2X)
  - 16GB of physical memory (300,000X) 64 bit VAs (48 active)
  - 2*300GB hard drives: silent, small and fixed (60,000X)
  - DVD±R: 4.7GB (30,000X)
  - Instructions retire every 300-500 picoseconds per CPU (8,000X faster per CPU * 2)

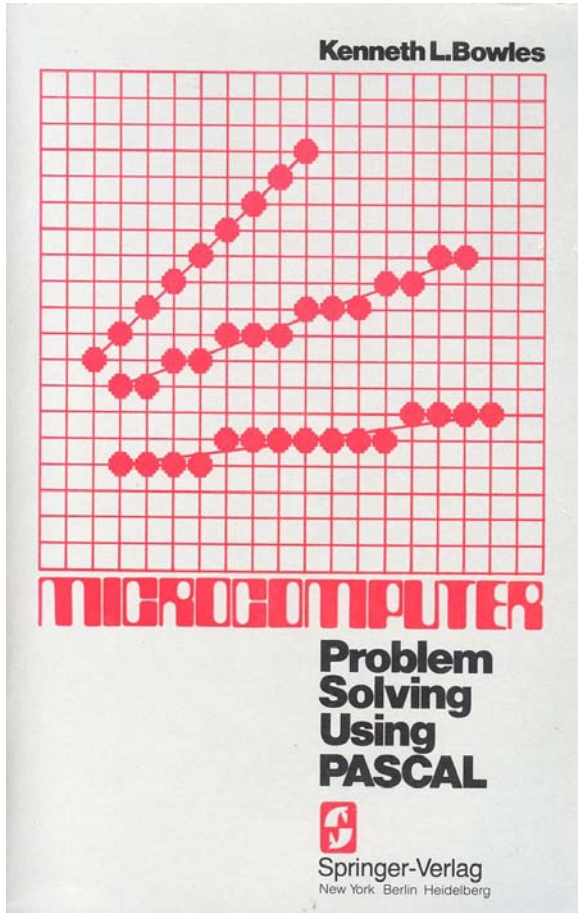
UCSD

Enter UCSD Pascal

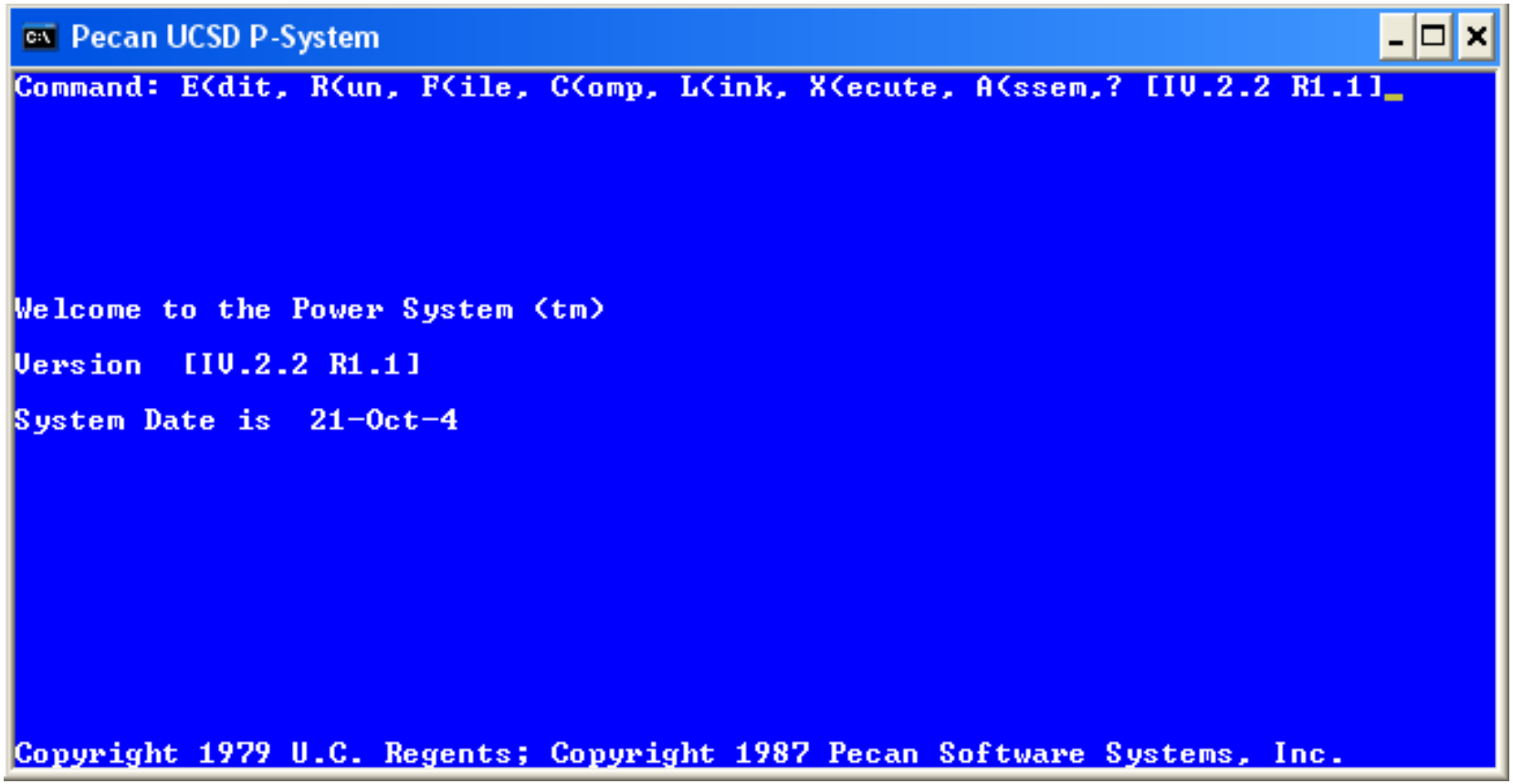
- UCSD needed a new intro programming course
  - Pascal instead of Burroughs Algol
    - Designed as a teaching language
    - Simple, expressive, encouraged good habits
  - Interactive instead of Batch
    - No arcane rituals (e.g. JCL)
    - Inexpensive computing resources
    - Instant feedback / gratification

Support for Courseware

- Edit-Compile-Run cycle highly tuned for students
  - Compiler or runtime error pops the user back into the Editor, showing where/why the problem occurred
    - StuPID bit controlled whether this was automatic, not one of our more politically correct moments
- Extensive set of quizzes developed for the intro programming course



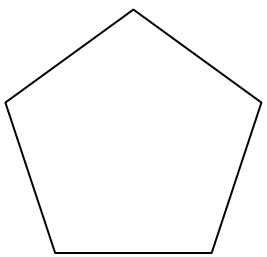
UCSD

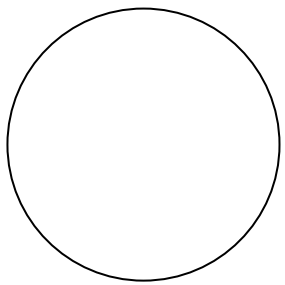

Pecan UCSD P-System
Command: E(dit, R(un, F(ile, C(omp, L(ink, X(ecute, A(ssem,? [IV.2.2 R1.1]
Welcome to the Power System (tm)
Version [IV.2.2 R1.1]
System Date is 21-Oct-4
Copyright 1979 U.C. Regents; Copyright 1987 Pecan Software Systems, Inc.

UCSD

Turtle Graphics

- Adapted Seymour Papert's LOGO
  - Turn(degree), TurnTo(degree)
  - Move(units), MoveTo(x,y)
  - PenColor(color)
- Great teaching tool
  - Intuitive
  - Rich & Expressive
  - Helped artsy types relate to computers

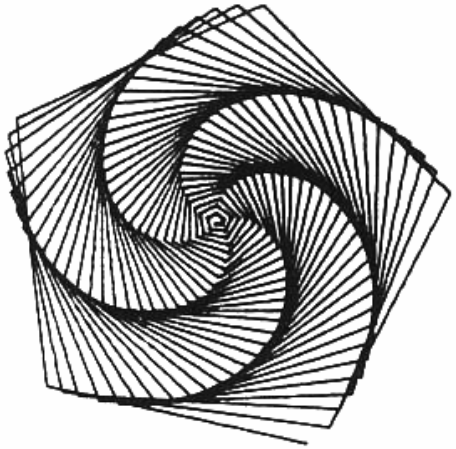


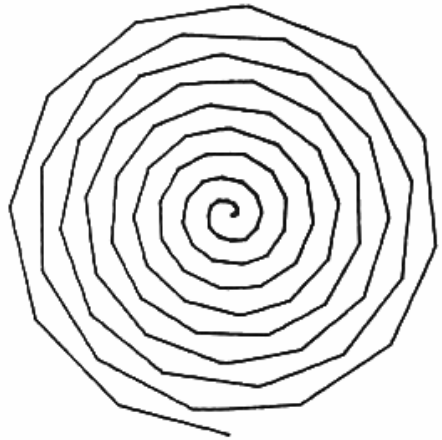


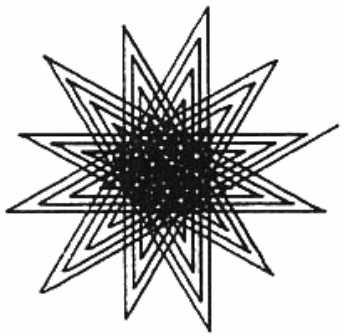
```
PROGRAM POLYGONS;
VAR SCALE: INTEGER;

PROCEDURE POLY(NSIDES,LGTH,
                ANGLE,X,Y:INTEGER);
VAR I: INTEGER;
BEGIN
  MOVETO(X*SCALE,Y*SCALE);
  PENCOLOR(WHITE);
  FOR I:=1 TO NSIDES DO
  BEGIN
    MOVE(LGTH*SCALE);
    TURN(ANGLE);
  END;
  PENCOLOR(NONE);
END;

BEGIN (* MAIN PROGRAM *)
  SCALE:=3; (* TERAK *)
  POLY(5,16,72,-30,-30);
  POLY(40,2,9,-30,6);
END.
```

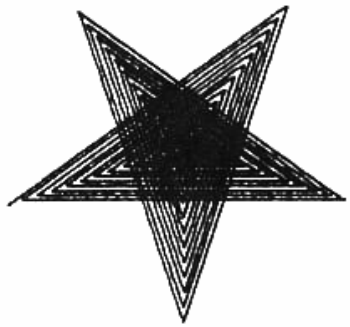

Source: "Microcomputer Problem Solving Using UCSD Pascal" (Ken Bowles 1977)

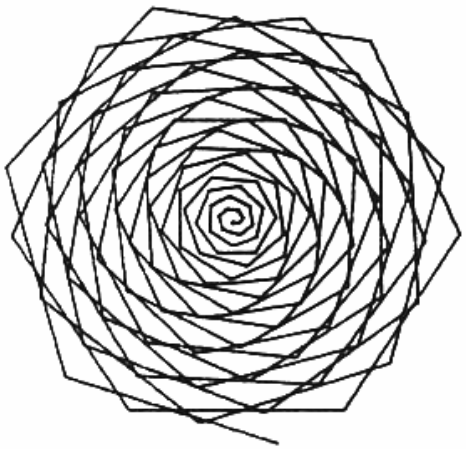
UCSD

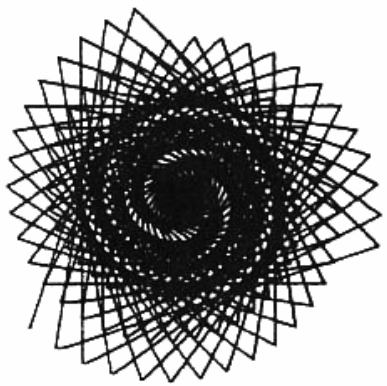








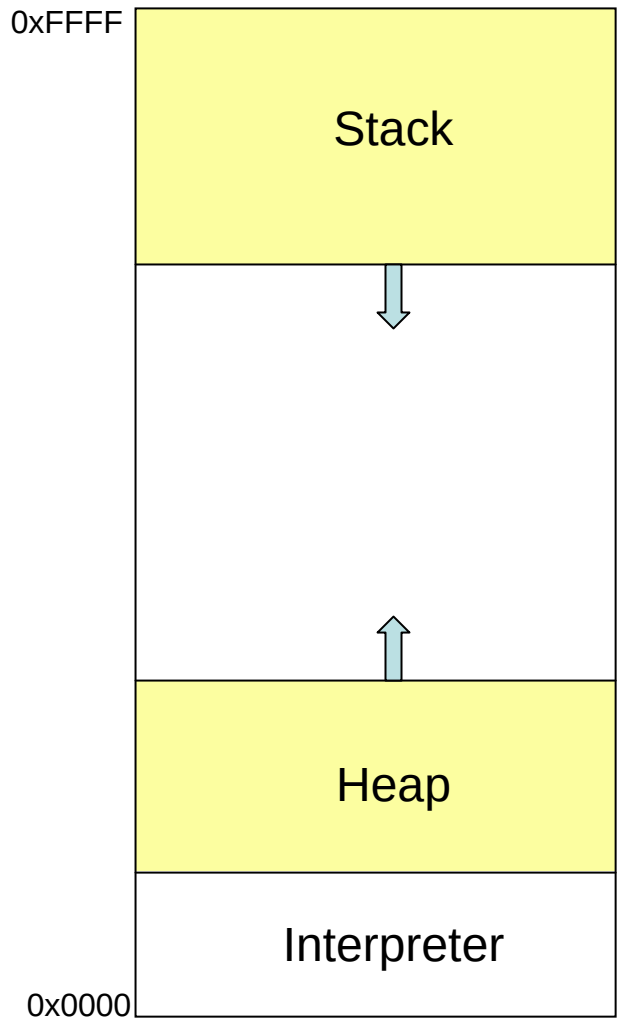




UCSD

Both a blessing and a curse: What could you fit in 64K?

- It sure kept the system clear of clutter!
  - ~10K of interpreter and raw device drivers
  - System + application code = ~20K - ~30K
    - Winword.exe is 12MB, not including shared libs. ~1,000X code bloat!
  - User stack + heap = the rest



UCSD

P-Code

- Extended compiler intermediate language from Urs Ammann's P2 compiler (ETH-Z)
  - With a little influence from Burroughs' B-Code
  - Variable-length instructions
    - Single-byte for the most common stack ops (e.g. push a small constant)
- Very slow (10X – 20X slower than native code)
  - Mitigated by intrinsics (e.g. byte move, scan and fill, reserved word lookup), and user patience
- PROCESSOR INDEPENDENT
  - The intellectual granddaddy of Java bytecode
  - Ported to PDP11, 8080/6, Z80, GA, TI, 6502, 6800 and ~10 others
  - Well-written code moved to other architectures without recompilation
    - "Endian-ness" issues (before it was even called that!)
- Very much a creature of 16-bit architectures

What About The System?

- Simple, simple, simple
  - Entire OS source was 2300 lines!
- Simple File System
  - Non-hierarchical directories
  - Contiguous files
    - User-initiated defrag ("K(runch")
  - 77 files per volume
  - 15 character filenames
  - Specialized text file format (compressed leading spaces)
- Simple memory management

Roger's License Plate

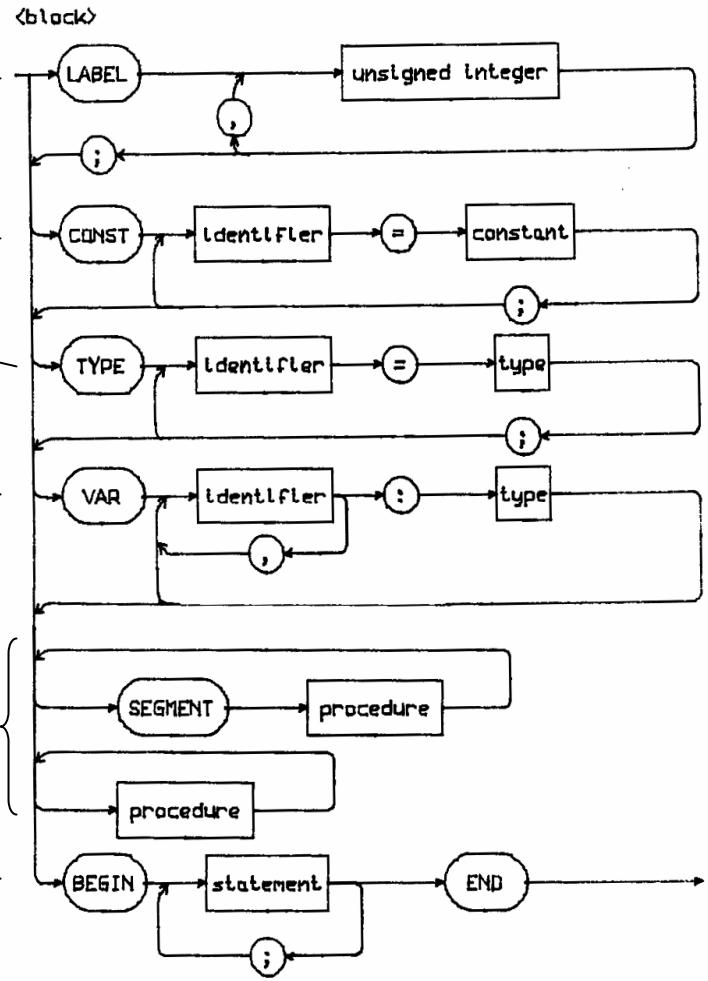
(Yes, back then it was yellow text on a blue background!)

UCSD

And the compiler…

- Used the ETH-Z P2 compiler as its base
- Recursive descent
  - Personal opinion: if a programming language can't be parsed simply, it doesn't need to live!
- Single pass – P-code emitted on the fly
  - Later incarnations had a "half-passed" fixup
- Simple enough transformation
  - Compiler was small and reliable
  - You could still watch it think, though!

```
BEGIN (*BLOCK*)
    REPEAT
      IF SY = LABELSY THEN
          BEGIN INSYMBOL; LABELDECLARATION END;
      IF SY = CONSTSY THEN
          BEGIN INSYMBOL; CONSTDECLARATION END;
      IF SY = TYPESY THEN
          BEGIN INSYMBOL; TYPEDECLARATION END;
      IF SY = VARSY THEN
          BEGIN INSYMBOL; VARDECLARATION END;
      WHILE SY IN [PROCSY,FUNCSY,PROGSY] DO
          BEGIN LSY := SY; INSYMBOL;
            IF LSY = PROGSY THEN SEGDECLARATION
            ELSE PROCDECLARATION(LSY,FALSE)
          END;
      IF SY <> BEGINSY THEN
          IF NOT (INCLUDING AND
              (SY IN [LABELSY,CONSTSY,TYPESY,VARSY,
                      PROCSY,FUNCSY,PROGSY])) THEN
            BEGIN ERROR(18); SKIP(FSYS) END
    UNTIL SY IN STATBEGSYS;
    DP := FALSE; IC := 0; LINEINFO := 0;
    IF SY = BEGINSY THEN INSYMBOL ELSE ERROR(17);
    IF NOT SYSCOMP THEN FINDFORW(DISPLAY[TOP].FNAME);
    REPEAT BODY(FSYS + [CASESY]);
      IF SY <> FSY THEN
          BEGIN ERROR(6); SKIP(FSYS + [FSY]) END
    UNTIL (SY = FSY) OR (SY IN BLOCKBEGSYS);
  END (*BLOCK*) ;
```



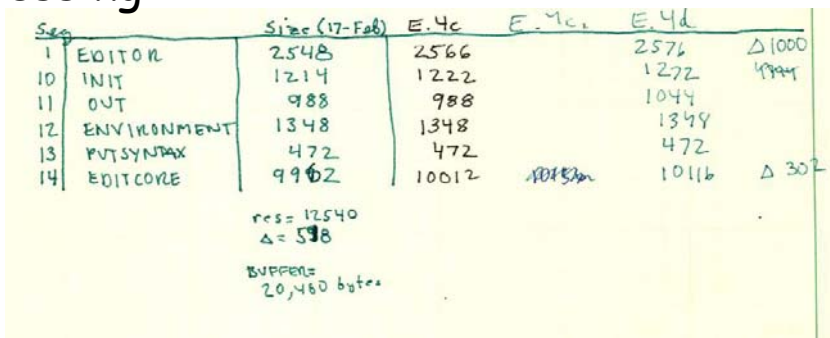
UCSD

Execution Environment

- "Segment procedures" used to control code residence
  - P-code loaded onto the stack from disk at call; code space reclaimed at exit
  - The user's program was really a segment procedure to the "real main program," the system.
- Mark/Release heap model
  - Mark(ptr) stored the current heap pointer, Release(ptr) cut the heap back to that point
  - Simple, but sharp edges!
    - Conservative automatic garbage collectors a little too rich for our blood!

The Screen Editor

- Consumed all spare time my junior and senior years
- In a world of 80 x 24 terminals, a WYSIWYG editor for flat text
  - Lots of help for Pascal programmers
    - Auto-indentation
  - A single, flat buffer to hold the source file
    - Y'all can guess how big a file it could edit!
    - Insertion: move everything to the right to make a hole!
- Later incarnations:
  - Handled arbitrarily large files, some fancier word processing features
    - Hand-in-hand. Zero-sum game between new features and maximum file size!
    - Nascent hypertext and macro processing
    - Outgrew its foundations
      - Not the only guilty party…



********** **********
* INTERACTIVE DEBUGGER * * Section 1.5 *
********** **********

Version II.0 February 1979

To facilitate the debugging of Pascal programs, an interactive debugger was included in the system in earlier releases. In order to use it, it required more memory than was available with any meaningfully sized program. We removed the debugger from the system as it was more of a thorn in the side of progress than a statement of progress itself. We are working on a new debugger and hope to have it in a useful state soon. The current changes in the P-machine may make the task of writing the debugger somewhat easier, and therefor quicker. Please do not inquire as to when the debugger will be ready for release, as the answer you will get will be "soon".

Thank-you for your patience and cooperation in this matter.

Ed.

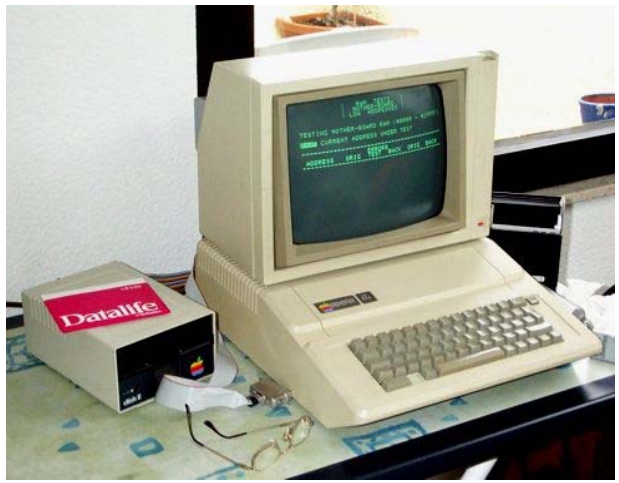
terak 8510/a GRAPHICS COMPUTER SYSTEM

- Workstation built around the LSI/11 chipset
  - 56K of memory (8K I/O space hurt!)
  - Single 8" 160K floppy
  - 320 x 240 B&W bitmap graphics
  - Keyboard layout customized for UCSD Pascal
- Very svelte in its day!
- Replaced the PDP 11/10s for the teaching labs

Source: The Terak Museum

Apple][c

- Based on a 1MHz Mostek 65C02
  - Our first units: 40 x 24!
    - OK if you didn't indent too much!
    - 64KB max
    - 5 ¼" floppies, ~140KB
- Replaced the Teraks
- Apple did a great job popularizing UCSD Pascal



http://www.s-line.de/homepages/horber-privat/apple2a.htm

UCSD

What Was Good About UCSD Pascal?

- It could host itself
  - The system, tools, applications and compiler were all written in Pascal
  - Bootstrapped from the B6700
    - Cutting the ties was WONDERFUL
- It was intuitive enough to use in introductory programming courses
- It was portable
  - This was a first. Unix didn't leave the PDP-11 until much later, and then not in binary
  - Rigorously separated porting vs. development effort

More Good Stuff

- Source was (at the start) freely distributed, and had a vibrant community of folks developing, extending and using it
- Raised the bar for micro-based software
  - Consistent, friendly user interface; smooth operation
- Starting point for Apple's][→ Lisa → Macintosh thrust
- Development platform for a Cray P-code to P-code optimizer
- Some interesting hardware developments
  - The Terak (more later)
  - NCR and Western Digital built custom processors for UCSD Pascal variants
  - Northwest Micro (Randy Bush), HP, many other had hardware packages optimized for UCSD Pascal
  - A line of Tektronix logic analyzers was built on top of UCSD Pascal
- **It was an amazing education for those of us on the project**

Pictures: Tektronix brochure, http://gallery.brouhaha.com/microengine

What Happened?

- Its virtues became its constraints
  - 16-bit flat addressing model for code+data
    - 64K → very, very simple code
    - No space for fancier software, larger problems, …
  - 16-bit flat address space couldn't effectively use the 8086's segmented memory model
    - MS-DOS could, even if you had to hold your nose
- Attempts at JIT native compilation didn't pan out
  - Valiant attempt
  - Even better performance required to fend off Borland Pascal (speed demon)
    - Renaissance System built a great 68K native compiler, but too late…
- We all graduated and went off to better things ☺
- The era of SofTech Microsystems
  - MarkO to talk about at the panel discussion

Credits

- John Foust, "The UCSD Pascal Museum" & "The Terak Museum" (http://www.threedee.com/jcm)