Commodore 64
Programming Languages:
Abacus
Super Pascal
ShadowM
World of Commodore 2014
speaker bio

- long-time collector of Commodore 64 compilers/interpreters
- a more-or-less neglected area for Commodore 64 enthusiasts
- hope to do a series of talks on some of the more interesting programming languages available
<table>
<thead>
<tr>
<th>common features</th>
</tr>
</thead>
<tbody>
<tr>
<td>does the language support...</td>
</tr>
<tr>
<td>• multiple drives?</td>
</tr>
<tr>
<td>• devices other than 8 and 9?</td>
</tr>
<tr>
<td>• writing large, modular programs?</td>
</tr>
<tr>
<td>• assembly language routines?</td>
</tr>
<tr>
<td>• standalone programs?</td>
</tr>
<tr>
<td>and, very important...</td>
</tr>
<tr>
<td>• does it have a good editor?</td>
</tr>
</tbody>
</table>
"historically correct" Pascal

- very much like UCSD p-System (virtual machine)
- early Pascal was meant to be portable across machines
- included low-level I/O primitives
- later concessions made to CP/M
not for the faint of heart

- replaces the C= operating system with its own, LOADDAT
- low-level format the same, but disk allocated in 4K blocks of 512-byte logical sectors (see section 7.3 of manual for details)
- requires SYSGEN to create work disks (which are bootable)
- not readable by C= DOS except for two stub entries in the directory
system disk... and work disk
so... is it worth using?

- MARK/RELEASE, NEW/DISPOSE
- integrated assembler; source can contain inline 6502 code!
- conditional compilation
- optional p-code output in listings
- "post-mortem" dumps
- "insert advice"
- memory/sector load, store, transfer
- hex dumps (memory and file)
- block table (view "BAM")
- internals very well documented, editor source is even provided
This is going to be FUN!
multiple drive support

- drives numbered 0 and 1
- compiler and assembler must be on drive 0 because of copy protection and overlays
- accessing drives other than 8 and 9 could probably be done in ML...
- disks from two drives can be merged to be treated as a single volume; "de-merging" them defrags the disk
merging disks

drive(map) = 0

map of disc "work1":
\textbf{loaddat}
disc 0 = 35 // blocks free!

drive(map) = 1

map of disc "work2":
\textbf{loaddat}
disc 0 = 35 // blocks free!

drive(map) = 0
disc Maybe used; sure to rewrite? y/y
disc-title = work
n of discs = 2

map of disc "work":
disc 0 = 39 // disc 1 = 39 // blocks free!
but how to copy to them...?

b(locktable) k(illtitle) t(rnsfrmem)
c(opy) l(ockfile) u(nlockfile)
d(uplicate) m(ap/drive) v(iewmem)
e(ntersect) n(ewdisc) w(rite)
f(etchsect) o(rganize) x(cludeblc)
g(etram) p(utram) y(listfile)
h(elp) q(uit) z(ero)
i(nsertadv) r(ename)

g start-adr. = $4000
file-title = loaddat
drive(map) = 0
end-adr.+i = $7fff

h
start-adr. = $4000
end-adr.+i = $7fff
file-title = loaddat
drive(map) = 0

map of disc "work  ":
loaddat
disc 0 = 35 // disc 1 = 39 //
blocks free!
OK, now what?

- merged disk boots successfully
- can't use for development because the compiler and assembler are copy-protected
- you could copy a program to it that needs to use a lot of disk space (i.e. a single large file)
- curiouser and curiouser...
modular programs

- SEGMENT keyword allows for up to eight overlays
- CONTINUE allows for program chaining
- EXECUTE allows for separate programs to be called as subroutines
- LOAD allows a module to be loaded at a specified address
assembly language support

- ML routines can also be assembled separately and used as modules
- supports conditional assembly
- object code address can be specified
- can also embed 6502 within Pascal source by using keyword ASSEMBLE
inline assembly

1000 program chtest;
1005 var c:char;
1010 (*---------------------------------------*)
1015 function chartest(testchar:char):char;
1020 assemble;
1025 chrourut .dl $ffd2
1030     php
1035     sei
1040     lda $01
1045     sta memmap
1050     lda #$37
1055     sta $01
1060     plp
1065     ldy #0
1070     lda (stkpoi),y
1075     jsr chrourut
1080     lda #$0d
1085     jsr chrourut
1090     inc stkpoi
1095     bne setret
1100     inc stkpoi+1
1105 setret lda #'y'
1110     ldy #0
1115     sta (stkpoi),y
1120     php
1125     sei
1130     lda memmap
1135     sta $01
1140     plp
1145     rts
1150 memmap .by 0
1160 (*---------------------------------------*)
1165 begin
1170     &pcode+;
1175     &adr+;
1180 writeln('Abacus Super Pascal inline ML test');
1185 writeln('character test: x');
1190     c:=chartest('x');
1195     if c='y' then
1200     writeln('returns y (OK)')
1205 else
1210     writeln('returns ',c,' (failed)');
1215     &adr-;
1220     &pcode-;
1225 end.
it runs!

```
commands = .
  a(ssembler)  h(elp)    r(unprgmr)
  c(ompiler)   j(ump)    u(tility)
  e(ditor)     m(ap/drive) w(ritesrcce)
  g(etraram)   p(utram)

 r
 file-title = chtest
drive(map) = 1
 Abacus Super Pascal inline ML test
ccharacter test: x
  x
returns y (OK)

ok
  * c=64  pascal-system  5.3 *
commands = .
  a(ssembler)  h(elp)    r(unprgmr)
  c(ompiler)   j(ump)    u(tility)
  e(ditor)     m(ap/drive) w(ritesrcce)
  g(etraram)   p(utram)
```

standalone programs?

- forget about it
how's the editor?

- old-school line-oriented editor; line numbers used while editing
- find/change/move/delete lines, auto-numbering/renumbering
- &CONTINUE and &INCLUDE for multiple source files
- formatted listings can be printed (from main menu, with WRITESRC)
- compiler automatically reloads source in editor after an error
additional features

- "post-mortem" runtime dumps
- "advice"
"post-mortem" dumps

- manual says "You'll be better off debugging the source-code, and just re-compiling the source."
- must enable as compile option
- asks at runtime whether to print
- dump itself not very readable
- no option to print again later?
program dump;
var i: integer;
begin
  &pcode+;
  &adr+;
  writeln('Super Pascal post-mortem dump test');
  writeln('4 div 2: ');
i := 4 div 2;
writeln('result is: ', i);
writeln('4 div 0: ');
i := 4 div 0;
writeln('result is: ', i);
&adr-;
&pcode-;
end.
compile options for dump

C

file-title = *
confirm "dump.p ,1"? n/y

* c=64 pascal-compiler 5.3 *

ready to compile: program "dump.p ,1"

default options ? n/n
start of prgm = $0800
start of heap = eopgm
top of stack = $9000
p.-code to disc ? n/y
tests of bounds ? n/y
suppress pmdump ? n/n
dump-title = p+m+dump
ignore a/p-opt.? n/n
suppress output ? n/n
suppr. hardcopy ? n/n
output device = 4,7

linking and saving "dump" ...... --
-> press: "return"
uh-oh! core dump!

4 div 2:
result is: 2
4 div 0:

zero-div. error in $08c3

* pascal post-mortem dump *

hardcopy of post-mortem dump? n/n
hex-list of array/record-var? n/y

the run-time error occurred
under the following conditions:

heap in usage from $08db upto $08db
stack in usage from $9000 downto $8fe2

executing program dump
at prgm-locality $08c3

variables:
i = 2 ($0002)
another way to find it...

1030  writeln('4 div 2:');

$088f  litw (3)
$0892  oprc (1)
$0893  oprc (1)
1035  writeln('4 div 2:');
$0894  litw (2)
$0896  litw (2)
$0898  oprc (1)
$0899  stow (2)
1040  writeln('result is: ',i);

$08a8  litw (3)
$08ab  oprc (1)
$08ac  lodw (2)
$08ae  oprc (1)
$08af  oprc (1)
1045  writeln('4 div 0:');

$08ba  litw (3)
$08bd  oprc (1)
$08be  oprc (1)
1050  writeln('4 div 0:');
$08bf  litw (2)
$08c1  litw (1)
$08c2  oprc (1)
$08c3  stow (2)
adding "advice"

help for: * c=64  file-utility  5.3 *

commands = ...
  a(device)  j(jump)  s(storemem)
  b(locktable)  k(illtitle)  t(rnsfrmem)
  c(opy)  l(lockfile)  u(nlockfile)
  d(uplicate)  m(ap/drive)  v(ietwmem)
  e(ntersect)  n(ewdisc)  w(rite)dir
  f(etchsect)  o(rganize)  x(cludeblk)
  g(etram)  p(utram)  y(listfile)
  h(elp)  q(uit)  z ero(block)
  i(nsertadv)  r(ename)

file-title = dump
confirm "dump .1"? n/y
write the advice (max. 63 char.)
and terminate with 'return'!
Purposely creates a runtime error and post-mortem dump.
...and reading it back

<table>
<thead>
<tr>
<th>Commands</th>
<th>Help for:</th>
</tr>
</thead>
<tbody>
<tr>
<td>a(device)</td>
<td>c=64 file-utility 5.3 *</td>
</tr>
<tr>
<td>b(locktable)</td>
<td></td>
</tr>
<tr>
<td>c(opy)</td>
<td></td>
</tr>
<tr>
<td>d(uplicate)</td>
<td></td>
</tr>
<tr>
<td>e(ntersect)</td>
<td></td>
</tr>
<tr>
<td>f(etchsect)</td>
<td></td>
</tr>
<tr>
<td>g(etram)</td>
<td></td>
</tr>
<tr>
<td>h(elp)</td>
<td></td>
</tr>
<tr>
<td>i(nsertadv)</td>
<td></td>
</tr>
<tr>
<td>s(toremem)</td>
<td></td>
</tr>
<tr>
<td>t(rnsfrmem)</td>
<td></td>
</tr>
<tr>
<td>u(nlockfile)</td>
<td></td>
</tr>
<tr>
<td>v(iewmem)</td>
<td></td>
</tr>
<tr>
<td>w(ritedir)</td>
<td></td>
</tr>
<tr>
<td>x(cludeblc)</td>
<td></td>
</tr>
<tr>
<td>y(listfile)</td>
<td></td>
</tr>
<tr>
<td>z(eroblock)</td>
<td></td>
</tr>
<tr>
<td>r(ename)</td>
<td></td>
</tr>
</tbody>
</table>

A file-title = dump
advice to "dump ,1":
Purposely creates a runtime error and po
st-mortem dump.
where to find it

- visit my site (www.lyonlabs.org) to get disk images and documentation:
  /commodore/onrequest/collections.html

- ZipCode disk images to re-create the original protected disk
- G64 image for VICE emulator
- PDF of original documentation