```
import "io"

manifest
{ sizeof_page =      0x800, // = 2048
  infopage    = 0xC0000000,
  intvec_va   = 0xC0000000,
  proclist_va = 0xC0000010,
  pgdir_va    = 0xC0000800,
  ptspec_va   = 0xC0001000,
  pgates_va   = 0xC0001800,
  proclist_max      = 32,
  proclist_max_bits = 5,
  ifp_freelistempty = 48,
  ifp_freelistptr = 49,
  ifp_freelistnum = 50,
  ifp_intvec_pa = 51,
  ifp_curproc = 52,
  p1_va    = 0xC0002000,   // virtual address for a handy page
  spec_p1e = 4,            // index in ptspec_va to put its PT entry
  p2_va    = 0xC0002800,
  spec_p2e = 5,
  p3_va    = 0xC0003000,
  spec_p3e = 6,
  p4_va    = 0xC0003800,
  spec_p4e = 7 }

export { infopage, intvec_va, ifp_freelistempty, ifp_freelistptr,
         ifp_freelistnum, ifp_intvec_pa, pgdir_va, ptspec_va,
         pgates_va, proclist_va, proclist_max, proclist_max_bits,
         ifp_curproc, p1_va, spec_p1e, p2_va, spec_p2e, p3_va,
         spec_p3e, p4_va, spec_p4e }

static { lastoccupiedpage, nextfreepage, firstnonpage }

let check_memory() be
{ let firstfreeword = ! 0x101;
  let lastexistingword = (! 0x100) - 1;
  lastoccupiedpage := (firstfreeword - 1) >> 11;
  nextfreepage := (firstfreeword + 2047) >> 11;
  firstnonpage := (lastexistingword >> 11) + 1;
  out("first free page = %d = 0x%x\n", nextfreepage, nextfreepage);
  out("last free page = %d = 0x%x\n", firstnonpage-1, firstnonpage-1); }

let getpage() be
{ let p = nextfreepage;
  nextfreepage +:= 1;
  if nextfreepage > firstnonpage then
  { outs("OUT OF MEMORY BEFORE VM STARTED\n");
    finish }
  resultis p }
```

```
let loados(name) be
{ let pgdir, ptcode, ptstack, ptspec, freelist, page, pinfopage, r, n,
      avail, pos, addr, pgates;
  r := devctl(DC_TAPE_LOAD, 1, name, 'R');
  if r < 0 then
  { out("error %d for loading tape '%s'\n", r, name);
    finish }

  pgdir := getpage() << 11;
  ptcode := getpage() << 11;
  ptstack := getpage() << 11;
  ptspec := getpage() << 11;
  pgates := getpage() << 11;

  assembly
  { clrpp [<pgdir>]
    clrpp [<ptcode>]
    clrpp [<ptstack>]
    clrpp [<ptspec>]
    clrpp [<pgates>]
    load  r1, [<pgates>]
    setsr r1, $cgbr }

  pgdir ! 0x200 := ptcode bitor 1;
  pgdir ! 0x2FF := ptstack bitor 1;
  pgdir ! 0x300 := ptspec bitor 1;
  ptstack ! 2047 := (getpage() << 11) bitor 1;
  page := getpage() << 11;
  ptcode ! 0 := page bitor 1;
  pinfopage := getpage() << 11;
  pinfopage ! ifp_intvec_pa := pinfopage;
  ptspec ! 0 := pinfopage bitor 1;
  ptspec ! 1 := pgdir bitor 1;
  ptspec ! 2 := ptspec bitor 1;
  ptspec ! 3 := pgates bitor 1;

  n := (firstnonpage + 2047) / 2048;
               // n = number of pages required to hold free page list
  for i = n-1 to 0 by -1 do
  { freelist := getpage() << 11;
    ptspec ! (2047 - i) := freelist bitor 1 }
  freelist +:= 2048;
               // = phys addr of page following last taken for free page list
  pinfopage ! ifp_freelistempty := 0xC0400000;
               // = virtual addr of location that would be immediately below
               //   bottom of free page list
```

```
  n := 0;
  avail := 8;
  pos := 1;
  addr := page + 0x400;
  while true do
  { r := devctl(DC_TAPE_READ, 1, addr);
    if r < 0 then
    { out("error %d while reading tape '%s'\n", r, name);
      finish }
    n +:= r;
    avail -:= 1;
    addr +:= 128;
    if r < 128 then break;
    if avail = 0 then
    { page := getpage() << 11;
      ptcode ! pos := page bitor 1;
      pos +:= 1;
      addr := page;
      avail := 16 } }
  out("%d bytes, %d pages of '%s' loaded\n", n, pos, name);

  n := 0;
  for i = firstnonpage-1 to nextfreepage by -1 do
  { freelist -:= 1;
    ! freelist := i;
    n +:= 1 }
  for i = lastoccupiedpage to 1 by -1 do
  { freelist -:= 1;
    ! freelist := i;
    n +:= 1 }
  pinfopage ! ifp_freelistptr := 0xC0400000 - n;
  pinfopage ! ifp_freelistnum := n;
  out("%d free pages in list\n", n);

  resultis pgdir }

let printmemmap(pdpn) be
{ let pdaddr = pdpn << 11;
  out("pp %d:\n", pdpn);    // pp = physical page
  for ptn = 0 to 1023 do
    if pdaddr ! ptn <> 0 then
    { let ptppn = (pdaddr ! ptn) >> 11;
      let ptaddr = ptppn << 11;
      out("  %d: pp %d for VAs 0x%x to 0x%x:\n",
             ptn,   ptppn,     ptn << 22, ((ptn+1) << 22)-1);
      for pn = 0 to 2047 do
        if ptaddr ! pn bitand 1 then
        { let pppn = (ptaddr ! pn) >> 11;
          let baseva = (ptn << 22) + (pn << 11);
          out("      %d: pp %d for VAs 0x%x to 0x%x:\n",
                     pn,    pppn,      baseva, baseva+2047); } } }
```

```
let start() be
{ let pdir;
  check_memory();
  pdir := loados("os2.exe");
  printmemmap(pdir >> 11);
  assembly
  { load   r1, [<pdir>]
    load   sp, 0x0000
    loadh  sp, 0xC000
    load   fp, sp
    setsr  r1, $pdbr
    getsr  r1, $flags
    sbit   r1, $vm
    load   r2, 0x0400
    loadh  r2, 0x8000
    flagsj r1, r2 }
  outs("Don't get here!\n"); }
```