```
-- file BuildNucleus.Mesa
-- last edited by Sandman, May 10, 1978  2:39 PM

DIRECTORY

  AltoDefs: FROM "altodefs",
  BcdDefs: FROM "bcddefs",
  BootCacheDefs: FROM "BootCacheDefs",
  BootmesaDefs: FROM "bootmesadefs",
  CommanderDefs: FROM "commanderdefs",
  ControlDefs: FROM "controldefs",
  FakeSegDefs: FROM "fakesegdefs",
  IODefs: FROM "iodefs",
  TimeDefs: FROM "timedefs",
  SegmentDefs: FROM "segmentdefs",
  StringDefs: FROM "stringdefs";

DEFINITIONS FROM BootmesaDefs;

BuildNucleus: PROGRAM
  IMPORTS BootmesaDefs, CommanderDefs, FakeSegDefs, IODefs, BootCacheDefs,
    SegmentDefs, StringDefs, TimeDefs
  EXPORTS BootmesaDefs =

  BEGIN

  FrameHandle: TYPE = ControlDefs.FrameHandle;
  GlobalFrameHandle: TYPE = ControlDefs.GlobalFrameHandle;

  name: STRING ← [40];

  Initialize: PROCEDURE =
    BEGIN OPEN TimeDefs, IODefs;
    timeofbuilding: STRING ← [18];
    version: BcdDefs.VersionStamp ← InitializeBootmesa[10,name];
    AppendDayTime[timeofbuilding,UnpackDT[version.time]];
    timeofbuilding.length ← timeofbuilding.length - 3;
    WriteChar[CR];
    WriteString[name]; WriteString[".Image -- "];  WriteLine[timeofbuilding];
    SetExtraFrames[];
    RETURN
    END;

  loadstateseg, initloadstateseg: FakeSegDefs.FakeSegmentHandle;
  bcdseg: FakeSegDefs.FakeSegmentHandle;

  RunCode: PROCEDURE =
    BEGIN OPEN SegmentDefs, BootmesaDefs;
    SwapInResidentCode[];
    FakeSegDefs.FakeSwapIn[initloadstateseg];
    FakeSegDefs.FakeSwapIn[loadstateseg];
    FakeSegDefs.FakeSwapIn[bcdseg];
    SwapInUserCode[];
    TurnOffStartTrap[];
    InitializeImage[];
    DeclareBootCommand[bc0];
    DeclareSegments[];
    DeclareVMBounds[FakeSegDefs.GetVMBounds[]];
    -- now we can write the image file
    WriteImage[];
    AdjustBcd[bcdseg];
    DeclareLoadStateParameters[loadstateseg, initloadstateseg, bcdseg];
    DeclareBootCommand[bc1];
    UnlockUserCode[];
    DeclareOpenFiles[]; -- lookup all requested files
    DeclareBootCommand[bc2];
    XFER[dest: WartFrame[]];
    RETURN
    END;

  SwapInResidentCode: PROCEDURE =
    BEGIN OPEN FakeSegDefs;
    codeseg: FakeSegmentHandle ← NIL;
    SwapInModule:
      PROCEDURE [frame: ControlDefs.GlobalFrameHandle] RETURNS [BOOLEAN] =
      BEGIN
```

```
      cseg: FakeSegmentHandle ← BootCacheDefs.READ[@frame.codesegment];
      IF cseg # codeseg THEN SwapInFrame[frame];
      IF codeseg = NIL THEN codeseg ← cseg;
      RETURN[FALSE]
      END;
    [] ← BootmesaDefs.EnumerateResidentModules[SwapInModule];
    END;

  GetFrames: PROCEDURE =
    BEGIN OPEN SegmentDefs;
    s: STRING ← [40];
    StringDefs.AppendString[s,name];
    StringDefs.AppendString[s,".bcd"];
    [loadstateseg, initloadstateseg, bcdseg] ← Load[s];
    SwapIn[bcdseg.Link2];
    InitUtilities[FileSegmentAddress[bcdseg.Link2]];
    BootmesaDefs.LookUpResidentModules[];
    END;

Find: PUBLIC PROCEDURE [module: STRING] RETURNS [frame: GlobalFrameHandle] =
  BEGIN
  frame ← Frame[module];
  IF frame = ControlDefs.NullGlobalFrame THEN
    BEGIN OPEN IODefs;
    WriteString["Can't find module "];
    WriteString[module];
    SIGNAL BootAbort;
    END;
  END;

BuildMesa: PROCEDURE =
  BEGIN
  OpenSource[name];
  ParseInput[];
  Initialize[];
  GetFrames[];
  RunCode[];
  CloseSource[];
  END;


-- main body code

BuildSystem: PROCEDURE [n: STRING] =
  BEGIN
  name.length ← 0;
  StringDefs.AppendString[name,n];
  BuildMesa[ !
    IODefs.Rubout => RETRY;
    BootAbort =>
      BEGIN
      IODefs.WriteLine[" Bootmesa aborted. Type any character to exit"];
      [] ← IODefs.ReadChar[];
      CONTINUE
      END];
  END;

CommanderDefs.AddCommand["Build", LOOPHOLE[BuildSystem], 1].params[0] ←
  [type: string, prompt: "System Name"];

END...
```