```
-- FindSigs.Mesa  Edited by Sandman on May 18, 1978  3:18 PM

DIRECTORY
  AltoDefs: FROM "altodefs",
  AltoFileDefs: FROM "altofiledefs",
  ControlDefs: FROM "controldefs",
  ImageDefs: FROM "imagedefs",
  ImageFileInfoDefs: FROM "imagefileinfodefs",
  InlineDefs: FROM "inlinedefs",
  IODefs: FROM "iodefs",
  MiscDefs: FROM "miscdefs",
  OutputDefs: FROM "outputdefs",
  SegmentDefs: FROM "segmentdefs",
  StreamDefs: FROM "streamdefs",
  StringDefs: FROM "stringdefs",
  SymbolTableDefs: FROM "symboltabledefs",
  SymDefs: FROM "symdefs",
  SystemDefs: FROM "systemdefs";

DEFINITIONS FROM AltoDefs, AltoFileDefs, SegmentDefs;

FindSigs: PROGRAM
  IMPORTS OutputDefs, SegmentDefs, StreamDefs, StringDefs, SymbolTableDefs,
    SystemDefs, ImageFileInfoDefs, MiscDefs =
  BEGIN

  nsigs: CARDINAL;

  SigItem: TYPE = RECORD [name: STRING, desc: CARDINAL];
  sigdata: ARRAY [0..128) OF SigItem;

  debugging: BOOLEAN ← FALSE;

  PrintSignals: PROCEDURE [
    symseg: FileSegmentHandle, name: STRING, gframe, gfi: CARDINAL] =
    BEGIN OPEN SymbolTableDefs, StreamDefs, SymDefs;
    tname: STRING ← [60];
    modname: STRING ← [60];
    ss: StringDefs.SubStringDescriptor;
    symbols: SymbolTableBase;
    sei: ISEIndex;
    modout: BOOLEAN ← FALSE;
    t, desc: CARDINAL;
    gfimask: CARDINAL ← 0;

    GetName: PROCEDURE [s: STRING, hti: HTIndex] =
      BEGIN
      symbols.SubStringForHash[@ss, hti];
      s.length ← 0;
      StringDefs.AppendSubString[s, @ss];
      END;

    LOOPHOLE[gfimask,ControlDefs.ProcDesc].gfi ← gfi;

    IF symseg = NIL THEN
      BEGIN OPEN OutputDefs;
      PutCR[];
      PutString[name]; PutString[" (cannot find symbols)"]; PutCR[];
      RETURN
      END;
    symbols ← AcquireSymbolTable[TableForSegment[symseg]];

    nsigs ← 0;
    WITH symbols.bb+FIRST[BTIndex] SELECT FROM
      Callable => GetName[modname, (symbols.seb+id).htptr];
      ENDCASE;
    FOR sei ← symbols.FirstCtxSe[symbols.stHandle.outerCtx],
      symbols.NextSe[sei] UNTIL sei = ISENull DO
      OPEN id: (symbols.seb+sei);
      IF id.writeonce THEN
        WITH (symbols.seb+symbols.UnderType[id.idtype]) SELECT FROM
          transfer =>
            IF (mode = signal OR mode = error) AND
              (symbols.seb+sei).ctxnum = symbols.stHandle.outerCtx THEN
              BEGIN
              GetName[tname, (symbols.seb+sei).htptr];
```

```
            desc ← (symbols.seb+sei).idvalue;
            t ← nsigs;
            WHILE t > 0 DO
              IF sigdata[t-1].desc < desc THEN EXIT;
              sigdata[t] ← sigdata[t-1];
              t ← t-1;
              ENDLOOP;
            nsigs ← nsigs+1;
            sigdata[t]← [SystemDefs.AllocateHeapString[tname.length], desc];
            StringDefs.AppendString[sigdata[t].name,tname];
            END;
        ENDCASE;
    ENDLOOP;

  IF nsigs > 0 THEN
    BEGIN OPEN OutputDefs;
    PutCR[];
    PutNumber[gframe, [8,FALSE,TRUE,6]];
    PutString["B "];
    PutString[modname]; PutCR[];
    FOR t IN [0..nsigs) DO
      PutString[" "];
      PutNumber[sigdata[t].desc+gfimask, [8,FALSE,TRUE,6]];
      PutString["B "];
      PutString[sigdata[t].name]; PutCR[];
      ENDLOOP;
    END;
  ReleaseSymbolTable[symbols];
  RETURN
  END;

ListSignals: PROCEDURE =
  BEGIN OPEN ImageFileInfoDefs;
  MungeModule: PROCEDURE [f: GlobalFrameHandle] RETURNS [BOOLEAN] =
    BEGIN
    BadName: PROCEDURE =
      BEGIN OPEN OutputDefs;
      PutString[name]; PutString[" (problems encountered)"L];
      END;
    BadFrame: PROCEDURE =
      BEGIN OPEN OutputDefs;
      PutOctal[f]; PutString[" (problems encountered)"L];
      END;
    seg: FileSegmentHandle;
    name.length ← 0;
    BEGIN
    FrameToModuleName[f, name ! ANY => BEGIN BadFrame[]; GOTO ret END];
    seg ← SymbolSegForFrame[f ! ANY => BEGIN BadName[]; GOTO ret END];
    PrintSignals[seg, name, LOOPHOLE[f], VirtualGlobalFrame[f].gfi !
      ANY => BEGIN BadName[]; CONTINUE END];
    EXITS ret => NULL;
    END;
    RETURN[FALSE]
    END;
  name: STRING ← [40];
  [] ← ImageFileInfoDefs.EnumerateGlobalFrames[MungeModule];
  OutputDefs.CloseOutput[];
  END;

CheckForExtension: PROCEDURE [name, ext: STRING] =
  BEGIN
  i: CARDINAL;
  FOR i IN [0..name.length) DO
    IF name[i] = '. THEN RETURN;
    ENDLOOP;
  StringDefs.AppendString[name, ext];
  RETURN
  END;

ProcessImage: PROCEDURE =
  BEGIN
  infile: STRING ← [40];
  root: STRING ← [40];
  i: CARDINAL;
  GetToken[infile];
  IF infile.length = 0 THEN SIGNAL Done;
```

```
    CheckForExtension[infile, ".image"];
    FOR i IN [0..infile.length) DO
      IF infile[i] = '. THEN EXIT;
      StringDefs.AppendChar[root, infile[i]];
      ENDLOOP;
    ImageFileInfoDefs.SetImage[infile];
    ImageFileInfoDefs.FindAllSymbols[];
    OutputDefs.OpenOutput[root, ".signals."L];
    WriteHerald[infile];
    ListSignals[];
    RETURN
    END;

  WriteHerald: PROCEDURE [name: STRING] =
    BEGIN OPEN OutputDefs;
    PutString[name];
    PutString[" -- "L];
    PutTime[ImageFileInfoDefs.Version[].time];
    PutCR[]; PutCR[]; PutCR[];
    RETURN
    END;

  GetToken: PROCEDURE [token: STRING] =
    BEGIN
    c: CHARACTER;
    token.length ← 0;
    UNTIL comstr.endof[comstr] DO
      SELECT c ← comstr.get[comstr] FROM
        IODefs.SP, IODefs.CR => IF token.length # 0 THEN RETURN;
        ENDCASE => StringDefs.AppendChar[token, c];
      ENDLOOP;
    RETURN
    END;

  GetCommandLineStream: PROCEDURE RETURNS [s: StreamDefs.StreamHandle] =
    BEGIN OPEN StreamDefs;
    cfa: POINTER TO AltoFileDefs.CFA ← MiscDefs.CommandLineCFA[];
    s ← CreateByteStream[SegmentDefs.InsertFile[@cfa.fp, Read], Read];
    JumpToFA[s, @cfa.fa];
    RETURN
    END;

  Done: SIGNAL = CODE;

  -- Main Body

  name: STRING ← [80];
  comstr: StreamDefs.StreamHandle ← GetCommandLineStream[];

  DO
    ProcessImage[ ! Done => EXIT];
    ENDLOOP;
  ImageDefs.StopMesa[];

  END...
```