```
-- file P5Debug.mesa
-- last modified by Sweet, June 30, 1978  5:28 PM

DIRECTORY
  CodeDefs: FROM "codedefs" USING [CCIndex, CCNull, ChunkBase, JumpCCIndex, JumpCCNull, JumpType],
  ErrorTabDefs: FROM "errortabdefs" USING [CompStrDesc],
  IODefs: FROM "iodefs" USING [CR, NumberFormat, WriteChar, WriteDecimal, WriteNumber, WriteOctal, Writ
**eString],
  P5ADefs: FROM "p5adefs" USING [ParamCount],
  P5BDefs: FROM "p5bdefs" USING [AreThreadsValid],
  SegmentDefs: FROM "segmentdefs" USING [DefaultPages, DefaultVersion, FileSegmentAddress, FileSegmentH
**andle, NewFile, NewFileSegment, Read, SwapIn, SwapOut, Unlock],
  StringDefs: FROM "stringdefs" USING [SubString, SubStringDescriptor],
  TableDefs: FROM "tabledefs" USING [AddNotify, DropNotify, TableNotifier],
  TreeDefs: FROM "treedefs" USING [treetype];

DEFINITIONS FROM CodeDefs;

P5Debug: PROGRAM
    IMPORTS IODefs, P5ADefs, P5BDefs, SegmentDefs, TableDefs =
  BEGIN

  cb: ChunkBase;                -- code base (local copy)

  P5DebugNotify: TableDefs.TableNotifier =
    BEGIN  -- called by allocator whenever table area is repacked
    cb ← LOOPHOLE[base[TreeDefs.treetype]];
    RETURN
    END;

  SubString: TYPE = StringDefs.SubString;
  fopnameSH: SegmentDefs.FileSegmentHandle ← NIL;
  mopnameSH: SegmentDefs.FileSegmentHandle ← NIL;
  fopbase, mopbase: STRING;

  fopnames: POINTER TO ARRAY [0..256) OF ErrorTabDefs.CompStrDesc;
  mopnames: POINTER TO ARRAY [0..256) OF ErrorTabDefs.CompStrDesc;

  SwapInOpTab: PROCEDURE =
    BEGIN OPEN SegmentDefs;
    fopbase ← mopbase ← NIL;
    BEGIN
      IF fopnameSH = NIL THEN fopnameSH ← NewFileSegment[
        NewFile["FOpNames.binary",Read,DefaultVersion !ANY => GOTO notfound],
        1, DefaultPages, Read];
      SwapIn[fopnameSH];
      fopnames ← FileSegmentAddress[fopnameSH]+1;
      fopbase ← LOOPHOLE[fopnames + 256*SIZE[ErrorTabDefs.CompStrDesc], STRING];
    EXITS notfound => NULL;
    END;
    BEGIN
      IF mopnameSH = NIL THEN mopnameSH ← NewFileSegment[
        NewFile["OpNames.binary",Read,DefaultVersion !ANY => GOTO notfound],
        1, DefaultPages, Read];
      SwapIn[mopnameSH];
      mopnames ← FileSegmentAddress[mopnameSH]+1;
      mopbase ← LOOPHOLE[mopnames + 256*SIZE[ErrorTabDefs.CompStrDesc], STRING];
    EXITS notfound => NULL;
    END;
    RETURN
    END;

  CloseOpTable: PROCEDURE =
    BEGIN
    IF fopnameSH # NIL THEN
      BEGIN
      SegmentDefs.Unlock[fopnameSH];
      SegmentDefs.SwapOut[fopnameSH];
      END;
    IF fopnameSH # NIL THEN
      BEGIN
      SegmentDefs.Unlock[mopnameSH];
      SegmentDefs.SwapOut[mopnameSH];
      END;
    RETURN
    END;
```

```
  WriteSubString: PROCEDURE [ss: SubString] =
    BEGIN
    i: CARDINAL;
    FOR i IN [ss.offset..ss.offset+ss.length)
      DO
      IODefs.WriteChar[ss.base[i]]
      ENDLOOP;
    RETURN
    END;

  PrintChunks: PUBLIC PROCEDURE [v: CodeDefs.CCIndex] =
    BEGIN
    OPEN IODefs;
    c: CCIndex;
    jc: JumpCCIndex;
    i: CARDINAL;
    ThreadsValid: BOOLEAN ← P5BDefs.AreThreadsValid[];

    TableDefs.AddNotify[P5DebugNotify];
    SwapInOpTab[];
    WriteChar[CR];
    FOR c ← v, cb[c].flink UNTIL c = CCNull DO
      WriteNumber[c, NumberFormat[10,FALSE,TRUE,6]];
      WITH cc:cb[c] SELECT FROM
        label =>
          BEGIN
          WriteString[": Label "L];
          IF ThreadsValid THEN
            FOR jc ← cc.jumplist, cb[jc].thread UNTIL jc = JumpCCNull DO
              WriteChar[' ]; WriteNumber[jc, NumberFormat[10,FALSE,TRUE,6]];
              ENDLOOP;
          END;
        jump =>
          BEGIN
          WriteString[": "L]; WriteJumpName[cc.jtype]; WriteChar[' ];
          WriteDecimal[LOOPHOLE[cc.destlabel]];
          END;
        code =>
          BEGIN
          WriteString[": Code "L];
          IF cc.realinst THEN WriteMOpName[cc.inst]
          ELSE WriteFOpName[cc.inst];
          FOR i IN [1..P5ADefs.ParamCount[LOOPHOLE[c]]] DO
            WriteNumber[cc.parameters[i], [8,FALSE,TRUE,7]];
            WriteChar['B];
            ENDLOOP;
          WriteString[", source "L]; WriteOctal[cc.sourcefileindex];
          END;
        other => WITH cc SELECT FROM
          table => WriteString[": Table"L];
          startbody, endbody =>
            BEGIN
            WriteString[IF cc.otag = startbody THEN ": Start"L ELSE ": End"L];
            WriteString["Body, bti: "L];
            WriteDecimal[LOOPHOLE[index]];
            END;
          ENDCASE;
        ENDCASE;
      IF cb[c].pad # 0 THEN WriteString[" pad=1"L];
      WriteChar[CR];
      ENDLOOP;
    CloseOpTable[];
    TableDefs.DropNotify[P5DebugNotify];
    RETURN
    END;

  WriteFOpName: PROCEDURE [n: CARDINAL] =
    BEGIN OPEN IODefs;
    opss: StringDefs.SubStringDescriptor;
    WriteChar['q];
    IF fopbase = NIL THEN WriteNumber[n,[8,TRUE,TRUE,3]]
    ELSE
      BEGIN
      opss.base ← fopbase;
```

```
    opss.offset ← fopnames[n].offset;
    opss.length ← fopnames[n].length;
    IF opss.length = 0 THEN WriteNumber[n,[8,TRUE,TRUE,3]]
    ELSE WriteSubString[@opss];
    END;
  RETURN
  END;

WriteMOpName: PROCEDURE [n: CARDINAL] =
  BEGIN OPEN IODefs;
  opss: StringDefs.SubStringDescriptor;
  WriteChar['z];
  IF mopbase = NIL THEN WriteNumber[n,[8,TRUE,TRUE,3]]
  ELSE
    BEGIN
    opss.base ← mopbase;
    opss.offset ← mopnames[n].offset;
    opss.length ← mopnames[n].length;
    IF opss.length = 0 THEN WriteNumber[n,[8,TRUE,TRUE,3]]
    ELSE WriteSubString[@opss];
    END;
  RETURN
  END;

WriteJumpName: PROCEDURE[n: CodeDefs.JumpType] =
  BEGIN OPEN IODefs;
  JumpName: ARRAY CodeDefs.JumpType OF STRING =
      ["JumpE"L, "JumpN"L, "JumpL"L, "JumpGE"L, "JumpG"L, "JumpLE"L,
       "UJumpL"L, "UJumpGE"L, "UJumpG"L, "UJumpLE"L, "ZJumpE"L, "ZJumpN"L,
       "Jump"L, "JumpA"L, "JumpC"L, "JumpCA"L, "JumpRet"L,
       "NILJumpE"L, "NILJumpN"L,
       "PAIRJumpL"L, "PAIRJumpG"L, "BYTEJumpE"L, "BYTEJumpN"L, "BITJumpE"L, "BITJumpN"L];

  IF n > LAST[JumpType] THEN
    BEGIN WriteChar['j]; WriteDecimal[LOOPHOLE[n]] END
  ELSE WriteString[JumpName[n]];  RETURN
  END;

END...
```