```
-- file CPass1.Mesa
-- last modified by Johnsson, July 25, 1978  4:55 PM

DIRECTORY
  AltoDefs: FROM "altodefs",
  ControlDefs: FROM "controldefs",
  SegmentDefs: FROM "segmentdefs",
  StringDefs: FROM "stringdefs",
  SymTabDefs: FROM "symtabdefs",
  SymDefs: FROM "symdefs",
  TableDefs: FROM "tabledefs",
  TreeDefs: FROM "treedefs";

CPass1: PROGRAM
    IMPORTS
        TableDefs, SymTabDefs =
  BEGIN
  OPEN SymTabDefs, SymDefs;

  -- from ComData
  -- basic types (initialized in Pass1)
    typeINTEGER, typeBOOLEAN, typeCHARACTER, typeSTRING: CSEIndex;
    typeREAL, typeLOCK, typeCONDITION: CSEIndex;

    idANY: ISEIndex;
    idINTEGER, idCARDINAL, idCHARACTER, idBOOLEAN, idSTRING: ISEIndex;
    idREAL, idLOCK: ISEIndex;

  -- anonymous entry for undeclared ids
    seAnon: ISEIndex;

  -- symbolic constants
    idTRUE, idFALSE: ISEIndex;

  -- global info describing module
    outerCtx: CTXIndex;          -- predefined identifiers

  -- symbol table bases
    seb: TableDefs.TableBase;    -- semantic entry base
    ctxb: TableDefs.TableBase;   -- context table base

  P1Notify: TableDefs.TableNotifier =
    BEGIN
    seb ← base[setype];  ctxb ← base[ctxtype];
    RETURN
    END;

 -- definition of standard symbols

  WordLength: CARDINAL = AltoDefs.wordlength;

  PrefillSymbols: PROCEDURE =
    BEGIN  -- called to prefill the compiler's symbol table
    tSei, ptrSei: CSEIndex;
    rSei: recordCSEIndex;
    tCtx: CTXIndex;
    sei: ISEIndex;
    outerCtx ← makenewctx[1Z];
    idANY ← MakeBasicType["UNSPECIFIED", codeANY, TRUE, WordLength];
      IF UnderType[idANY] # typeANY THEN ERROR;
    idINTEGER ← MakeBasicType["INTEGER", codeINTEGER, TRUE, WordLength];
      typeINTEGER ← UnderType[idINTEGER];
    idCHARACTER ← MakeBasicType["CHARACTER", codeCHARACTER, TRUE, AltoDefs.charlength];
      typeCHARACTER ← UnderType[idCHARACTER];
    -- make BOOLEAN type
      typeBOOLEAN ← makenonctxse[SIZE[enumerated constructor SERecord]];
      idBOOLEAN ← MakeNamedType["BOOLEAN", typeBOOLEAN];
      tCtx ← makenewctx[1Z];
      (seb+typeBOOLEAN)↑ ← SERecord[mark3: TRUE, mark4: TRUE,
        sebody: constructor[
            enumerated[
                ordered: TRUE,
                valuectx: tCtx,
                nvalues: 2]]];
      [] ← MakeConstant["FALSE", tCtx, idBOOLEAN, 0];
      [] ← MakeConstant["TRUE", tCtx, idBOOLEAN, 1];
```

```
    resetctxlist[tCtx];
  idCARDINAL ← MakeSubrangeType["CARDINAL", 0, AltoDefs.maxword];
  [] ← MakeNamedType["WORD", UnderType[idCARDINAL]];
  -- make REAL type
    typeREAL ← makenonctxse[SIZE[real constructor SERecord]];
    (seb+typeREAL)↑ ← SERecord[mark3: TRUE, mark4: TRUE,
      sebody: constructor[real[rangetype: idINTEGER]]];
    idREAL ← MakeNamedType["REAL", typeREAL];
  -- make STRING type
    rSei ← MakeRecord[nFields:3, nBits:2*WordLength];
    [] ← MakeField["length", idCARDINAL, [wd:0, bd:0], WordLength];
    sei ← MakeField["maxlength", idCARDINAL, [wd:1, bd:0], WordLength];
    (seb+sei).writeonce ← TRUE;
    tSei ← makenonctxse[SIZE[array constructor SERecord]];
    (seb+tSei)↑ ← SERecord[mark3: TRUE, mark4: TRUE,
      sebody: constructor[
          array[
              packed: TRUE,
              indextype: idCARDINAL,  -- a fudge
              componenttype: idCHARACTER,
              comparable: FALSE,
              lengthUsed: FALSE]]];
    sei ← MakeField["text", tSei, [wd:2, bd:0], 0];
    tSei ← MakePointerType[MakeNamedType["StringBody", rSei]];
    idSTRING ← MakeNamedType["STRING", tSei];
    typeSTRING ← UnderType[idSTRING];
  -- make LOCK type
    rSei ← MakeRecord[nFields:1, nBits:WordLength];
    (seb+rSei).unifield ← FALSE;
    [] ← MakeField[NIL, idANY, [wd:0, bd:0], WordLength];
    idLOCK ← MakeNamedType["MONITORLOCK", rSei];
    typeLOCK ← UnderType[idLOCK];
  -- make CONDITION type
    rSei ← rSei ← MakeRecord[nFields:2, nBits:2*WordLength];
    [] ← MakeField[NIL, idANY, [wd:0, bd:0], WordLength];
    [] ← MakeField["timeout", idCARDINAL, [wd:1, bd:0], WordLength];
    typeCONDITION ← UnderType[MakeNamedType["CONDITION", rSei]];
  -- make a universal pointer type
    ptrSei ← MakePointerType[typeANY];
  -- enter the Boolean constants
    idTRUE ← MakeConstant["TRUE", outerCtx, idBOOLEAN, 1];
    idFALSE ← MakeConstant["FALSE", outerCtx, idBOOLEAN, 0];
  -- make a universal NIL
    [] ← MakeConstant["NIL", outerCtx, ptrSei, 0];
  -- make a neutral entry for error recovery
    seAnon ← MakeVariable[
      name: "?",
      ctx: outerCtx,
      type: typeANY,
      offset: [wd:0, bd:0],
      nBits: WordLength];
  -- predeclare UNWIND
    tSei ← makenonctxse[SIZE[transfer constructor SERecord]];
    (seb+tSei)↑ ← SERecord[mark3: TRUE, mark4: TRUE,
      sebody: constructor[
          transfer[
              mode: error,
              inrecord: recordCSENull,
              outrecord: recordCSENull]]];
    [--idUNWIND--] ← MakeConstant["UNWIND", outerCtx, tSei,
          ControlDefs.ControlLink[procedure[
              gfi: ControlDefs.GFTNull,
              ep: ControlDefs.EPRange-1,
              tag: procedure]]];
  -- make some constants
--    BEGIN
--    tC0 ← [literal[word[index: LitDefs.FindLiteral[0]]]];
--    tC1 ← [literal[word[index: LitDefs.FindLiteral[1]]]];
--    END;
  resetctxlist[outerCtx];
  RETURN
  END;


SubStringDescriptor: TYPE = StringDefs.SubStringDescriptor;
```

```
MakeNamedType: PROCEDURE [s: STRING, type: SEIndex] RETURNS [sei: ISEIndex] =
  BEGIN
  desc: SubStringDescriptor ← [base:s, offset:0, length:s.length];
  sei ← makectxse[EnterString[@desc], outerCtx];
    BEGIN  OPEN (seb+sei);
    idtype ← typeTYPE;  idinfo ← type;  idvalue ← TreeDefs.empty;
    writeonce ← constant ← TRUE;
    extended ← public ← linkSpace ← FALSE;
    mark3 ← mark4 ← TRUE;
    END;
  RETURN
  END;

MakeBasicType: PROCEDURE
      [s: STRING, code: [0..16), ordered: BOOLEAN, nBits: CARDINAL]
      RETURNS [ISEIndex] =
  BEGIN  -- makes an se entry for a built-in type --
  sei: CSEIndex = makenonctxse[SIZE[basic constructor SERecord]];
  (seb+sei)↑ ← SERecord[mark3: TRUE, mark4: TRUE,
      sebody: constructor[
          basic[ordered:ordered, code:code, length:nBits]]];
  RETURN [MakeNamedType [s, sei]]
  END;

MakeConstant: PROCEDURE
      [name: STRING, ctx: CTXIndex, type: SEIndex, value: UNSPECIFIED]
      RETURNS [sei: ISEIndex] =
  BEGIN  -- makes an se entry for a built-in constant --
  desc: SubStringDescriptor ← [base:name, offset:0, length:name.length];
  sei ← makectxse[EnterString[@desc], ctx];
    BEGIN  OPEN (seb+sei);
    idtype ← type;  idinfo ← 0;  idvalue ← value;
    writeonce ← constant ← TRUE;
    extended ← public ← linkSpace ← FALSE;
    mark3 ← mark4 ← TRUE;
    END;
  RETURN
  END;

MakeVariable: PROCEDURE
      [name: STRING, ctx: CTXIndex, type: SEIndex, offset: BitAddress, nBits: CARDINAL]
      RETURNS [sei: ISEIndex] =
  BEGIN
  desc: SubStringDescriptor ← [base:name, offset:0, length:name.length];
  sei ← makectxse[EnterString[@desc], ctx];
    BEGIN  OPEN (seb+sei);
    idtype ← type;  idvalue ← offset;  idinfo ← nBits;
    writeonce ← constant ← public ← extended ← linkSpace ← FALSE;
    mark3 ← mark4 ← TRUE;
    END;
  RETURN
  END;


rCtx: CTXIndex;
seChain: ISEIndex;

MakeRecord: PROCEDURE [nFields, nBits: CARDINAL] RETURNS [rSei: recordCSEIndex] =
  BEGIN
  rSei ← LOOPHOLE[makenonctxse[SIZE[notlinked record constructor SERecord]]];
  rCtx ← makenewctx[1Z];
  (ctxb+rCtx).selist ← seChain ← makeSEChain[rCtx, nFields, FALSE];
  (seb+rSei)↑ ← SERecord[mark3: TRUE, mark4: TRUE,
      sebody: constructor[
          record[
              machineDep: TRUE,
              unifield: nFields = 1,
              argument: FALSE,
              defaultFields: FALSE,
              fieldctx: rCtx,
              length: nBits,
              comparable: FALSE,
              privateFields: FALSE,
              lengthUsed: FALSE,
              monitored: FALSE,
              variant: FALSE,
```

```
            linkpart: notlinked[]]]];
  RETURN
  END;

MakeField: PROCEDURE
      [name: STRING, type: SEIndex, offset: BitAddress, nBits: CARDINAL]
      RETURNS [sei: ISEIndex] =
  BEGIN
  desc: SubStringDescriptor;
  hti: HTIndex;
  IF name # NIL
    THEN
      BEGIN
      desc ← [base:name, offset:0, length:name.length];
      hti ← EnterString[@desc];
      END
    ELSE hti ← HTNull;
  sei ← seChain;  seChain ← NextSe[seChain];
  fillctxse[sei, hti, FALSE];
    BEGIN  OPEN (seb+sei);
    idtype ← type;  idvalue ← offset;  idinfo ← nBits;
    writeonce ← constant ← public ← extended ← linkSpace ← FALSE;
    mark3 ← mark4 ← TRUE;
    END;
  RETURN
  END;

MakePointerType: PROCEDURE [refType: SEIndex] RETURNS [sei: CSEIndex] =
  BEGIN
  sei ← makenonctxse[SIZE[pointer constructor SERecord]];
  (seb+sei)↑ ← SERecord[mark3: TRUE, mark4: TRUE,
      sebody: constructor[
          pointer[
              ordered: FALSE,
              readonly: FALSE,
              basing: FALSE,
              pointedtotype: refType,
              dereferenced: TRUE]]];
  RETURN
  END;

MakeSubrangeType: PROCEDURE
      [s: STRING, origin: INTEGER, range: CARDINAL]
      RETURNS [ISEIndex] =
  BEGIN
  sei: CSEIndex;
  sei ← makenonctxse[SIZE[subrange constructor SERecord]];
  (seb+sei)↑ ← SERecord[mark3: TRUE, mark4: TRUE,
      sebody: constructor[
          subrange[
              filled: TRUE,
              empty: FALSE,
              flexible: FALSE,
              rangetype: idINTEGER,
              origin: origin,
              range: range]]];
  RETURN [MakeNamedType[s, sei]]
  END;


P1Unit: PUBLIC PROCEDURE =
  BEGIN  OPEN SegmentDefs;
  TableDefs.AddNotify[P1Notify];
  PrefillSymbols[];
  TableDefs.DropNotify[P1Notify];
  RETURN
  END;

END.
```