```
-- file codedefs.mesa
-- last modified by Sweet, July 14, 1978  2:17 PM

DIRECTORY
  AltoDefs: FROM "altodefs" USING [Address, BYTE, wordlength],
  LitDefs: FROM "litdefs" USING [LTIndex, STIndex],
  SymDefs: FROM "symdefs" USING [BTIndex, ByteIndex, ContextLevel, HTIndex, ISEIndex, ISENull, lZ],
  TableDefs: FROM "tabledefs" USING [TableLimit, TableNotifier];

DEFINITIONS FROM LitDefs, TableDefs, SymDefs;

CodeDefs: DEFINITIONS =
  BEGIN

  Lexeme: TYPE = RECORD [
        lexvalue: SELECT lextag: * FROM
          literal => [
                SELECT littag: * FROM
                  word => [lexlti: LTIndex],
                  string => [lexsti: STIndex],
                  ENDCASE],
          se => [lexsei: ISEIndex],
          bdo => [lexbdoi: BDOIndex],
          other => [
            SELECT sublextag: * FROM
              register => [lexrn: RegisterName],
              byte => [lexalpha: ByteOffset, long: BOOLEAN],
              ENDCASE],
          ENDCASE];


  topostack: se Lexeme = Lexeme[lexvalue: se[ISENull]];

  RegisterName: TYPE = [0..17777B]; -- fill variant record to 16 bits
  ByteOffset: TYPE = [0..7777B]; -- fill variant record to 16 bits

  ChunkBase: TYPE = BASE POINTER; -- to chunk area of compiler data space

  BDOIndex: TYPE = ChunkBase RELATIVE POINTER [0..TableLimit) TO BDOItem;
  BDONull: BDOIndex = LOOPHOLE[TableLimit-1];
  InUseThread: BDOIndex = LOOPHOLE[TableLimit-2];

  BDOComponentNames: TYPE = {basecomponent, dispcomponent, offsetcomponent};

  BDOComponent: TYPE = RECORD [
        posn: FullBitAddress,
        size: WORD,
        level: ContextLevel];

  BDOItem: TYPE = RECORD [
        free: BOOLEAN,
        thread: BDOIndex,
        tag: BDOTag,
        base: BDOComponent,
        disp: BDOComponent,
        offset: BDOComponent];

  BDOTag: TYPE = {bdo, bo, o};

  FullBitAddress: TYPE = RECORD [
        wd: AltoDefs.Address, bd: [0..AltoDefs.wordlength)];

  lTOS: ContextLevel = LAST[ContextLevel];

  TosBDOComponent: BDOComponent =
    BDOComponent[level: lTOS, posn: FullBitAddress[0, 0], size: AltoDefs.wordlength];
  WordZeroBDOComponent: BDOComponent =
    BDOComponent[level: lZ, posn: FullBitAddress[0, 0], size: AltoDefs.wordlength];

  CodeChunkType: TYPE = {code, label, jump, other};

  CCItem: TYPE = RECORD [
        free: BOOLEAN,
        pad: [0..1], -- this is NOT a fill field
        flink, blink: CCIndex,
        ccvalue: SELECT cctag: CodeChunkType FROM
        code => [
              sourcefileindex: ByteIndex,
              realinst, minimalStack: BOOLEAN,
              inst: AltoDefs.BYTE,
              aligned: BOOLEAN,
              isize: [0..3],
              fill: [0..17B),
              parameters: ARRAY [1..1) OF WORD],
        label => [
              labelseen: BOOLEAN,
              jumplist: JumpCCIndex],
        jump => [
              jsize: [0..7],
              jtype: JumpType,
              jparam: AltoDefs.BYTE,
              forward: BOOLEAN,
              thread: JumpCCIndex,
              jbytes: INTEGER,
              fixedup, completed: BOOLEAN,
              destlabel: LabelCCIndex],
        other => [obody: SELECT otag: * FROM
          table => [
              btab: BOOLEAN,
              tablecodebytes: [0..7],
              taboffset: INTEGER],
          startbody, endbody => [
              index: BTIndex],
          ENDCASE],
        ENDCASE];

  NULLfileindex: ByteIndex = -1;

  CCIndex: TYPE = ChunkBase RELATIVE POINTER [0..TableLimit) TO CCItem;
  CCNull: CCIndex = LOOPHOLE[TableLimit-1];
  JumpCCIndex: TYPE = ChunkBase RELATIVE POINTER [0..TableLimit) TO jump CCItem;
  JumpCCNull: JumpCCIndex = LOOPHOLE[TableLimit-1];
  LabelCCIndex: TYPE = ChunkBase RELATIVE POINTER [0..TableLimit) TO label CCItem;
  LabelCCNull: LabelCCIndex = LOOPHOLE[TableLimit-1];
  CodeCCIndex: TYPE = ChunkBase RELATIVE POINTER [0..TableLimit) TO code CCItem;
  CodeCCNull: CodeCCIndex = LOOPHOLE[TableLimit-1];
  OtherCCIndex: TYPE = ChunkBase RELATIVE POINTER [0..TableLimit) TO other CCItem;
  TableCCIndex: TYPE = ChunkBase RELATIVE POINTER [0..TableLimit) TO table other CCItem;
  TableCCNull: TableCCIndex = LOOPHOLE[TableLimit-1];

  CompareClass: TYPE = {word, byte};

  JumpType: TYPE =
        {JumpE, JumpN, JumpL, JumpGE, JumpG, JumpLE,
         UJumpL, UJumpGE, UJumpG, UJumpLE, ZJumpE, ZJumpN,
         Jump, JumpA, JumpC, JumpCA, JumpRet,
         NILJumpE, NILJumpN, PAIRJumpL, PAIRJumpG,
         BYTEJumpE, BYTEJumpN, BITJumpE, BITJumpN};

  EXLabelRecord: TYPE = RECORD [
        free: BOOLEAN,
        thread: EXLRIndex,
        labelcc: CARDINAL,
        labelhti: HTIndex,
        labelcci: LabelCCIndex];

  EXLRIndex: TYPE = ChunkBase RELATIVE POINTER [0..TableLimit) TO EXLabelRecord;
  EXLRNull: EXLRIndex = LOOPHOLE[TableLimit-1];

  StkItem: TYPE = RECORD[
        uplink,downlink: StkIndex,
        stkvalue: SELECT stktag: * FROM
          item => [lexeme: se Lexeme],
          MARK => [label: LabelCCIndex],
          ENDCASE];

  StkIndex: TYPE = POINTER TO StkItem;

  EvalStackSize: INTEGER = 8;
  MaxParmsInStack: INTEGER = EvalStackSize-3;

  TempStateRecord: TYPE = RECORD[
      pendtemplist, templist, heaplist: ISEIndex,
      tempctxlvl: ContextLevel,
      tempstart, framesz: INTEGER];

 ChunkIndex: TYPE = ChunkBase RELATIVE POINTER [0..TableDefs.TableLimit);

 GetChunk: PROCEDURE [size: CARDINAL] RETURNS [ChunkIndex];
 FreeChunk: PROCEDURE [i: ChunkIndex, size: CARDINAL];

 DriverNotify, AddressNotify, StackNotify, FlowNotify, StoreNotify,
      ExpressionNotify, FlowExpressionNotify, StatementNotify, CallsNotify,
      OutCodeNotify, PeepholeNotify, JumpsNotify, FinalNotify:
        TableDefs.TableNotifier;

 OpTable, Driver, Address, Stack, Flow, Store, Expression, FlowExpression,
      Statement, Calls, OutCode, PeepholeQ, PeepholeU, PeepholeZ,
      Jumps, Final:
        PROGRAM;

END...
```