```
begin
  block number = 0 LOCCTR[i] = 0 for all i
  read the first input line
  if OPCODE = 'START' then
  begin
    write line to intermediate file
    read next input line
  end {if START}
  while OPCODE ≠ 'END' do
  if OPCODE = 'USE'
  begin
    if there is no OPEREND name then
      set block name as default
    else block name as OPERAND name
    if there is no entry for block name then
      insert (block name, block number ++) in block table
    i = block number for block name
    if this is not a comment line then
      begin
      if there is a symbol in the LABEL field then
        begin
        search SYMTAB for LABEL
        if found then
          set error flag (duplicate symbol)
        else
          insert (LABEL, LOCCTR[i]) into SYMTAB
        end {if symbol}
      Search OPTAB for OPCODE
      if found then
        add 3 instruction length to LOCCTR[i]
      else if OPCODE = 'WORD' then
        add 3 to LOCCTR[i]
      else if OPCODE = 'RESW' then
        add 3 * #[OPERAND] to LOCCTR[i]
      else if OPCODE = 'RESB' then
        add #[OPERAND] to LOCCTR[i]
      else if OPCODE = 'BYTE' then
      begin
        find length of constant in bytes
        add length to LOCCTR[i]
      end {if byte}
  else
```

Figure 2.12(b) Pass 1 of program blocks.

```
    Set error flag
    end {if not a comment}
  write line to intermediate file
  read Text input line
  end {while not END}
write last line to intermediate file
save Length[i] as LOCCTR[i] for all i
Address[o] = starting address
Address[i] = address(i - 1) + Length(i - 1)
             [for i = 1 to max(block number)]
insert(address[i], Length[i]) in block table for all i
end {Pass 1}
```

Figure 2.12(b) (*cont'd*)

```
If OPCODE = 'USE' then
  set block number for block name with OPERAND field
  search SYMTAB for OPERAND
  store symbol value + address [block number] as operand address
end {Pass 2}
```

Figure 2.12(c) Pass 2 of program blocks.