```
1              START    200
2              MOVER    AREG, ='5'
3              MOVEM    AREG, A
4    LOOP      MOVER    AREG, A
5              MOVER    CREG, B
6              ADD      CREG, ='1'
7              ...

12             BC       ANY, NEXT
13             LTORG
                        ='5'      #1
                        ='1'

14             ...
15   NEXT      SUB      AREG, ='1'
16             BC       LT, BACK
17   LAST      STOP
18             ORIGIN   LOOP+2
19             MULT     CREG, B
20             ORIGIN   LAST+1
21   A         DS       1
22   BACK      EQU      LOOP
23   B         DS       1
24             END
25                      ='1'
```

Algorithm 4.1 (Assembler First Pass)

```
1. loc_cntr := 0; (default value)
   pooltab_ptr := 1; POOLTAB [1] := 1;
   littab_ptr := 1;
2. While next statement is not an END statement          RC = 0
   (a) If label is present then                          RC = 0
           this_label := symbol in label field;
           Enter (this_label, loc_cntr) in SYMTAB.
   (b) If an LTORG statement then
       (i) Process literals LITTAB [POOLTAB [pooltab_ptr]] ... LITTAB [littab_ptr−1] to allocate memory and put the address in the address field. Update loc_cntr accordingly.
       (ii) pooltab_ptr := pooltab_ptr + 1;
       (iii) POOLTAB [pooltab_ptr] := littab_ptr;
   (c) If a START or ORIGIN statement then
           loc_cntr := value specified in operand field;
   (d) If an EQU statement then
       (i) this_addr := value of <address spec>;
       (ii) Correct the symtab entry for this_label to (this_label, this_addr).
   (e) If a declaration statement then
       (i) code := code of the declaration statement;
       (ii) size := size of memory area required by DC/DS.
       (iii) loc_cntr := loc_cntr + size;
       (iv) Generate IC '(DL, code) ···'.
   (f) If an imperative statement then
       (i) code := machine opcode from OPTAB;
       (ii) loc_cntr := loc_cntr + instruction length from OPTAB;
       (iii) If operand is a literal then
                 this_literal := literal in operand field;
                 LITTAB [littab_ptr] := this_literal;
                 littab_ptr := littab_ptr + 1;
             else (i.e. operand is a symbol)
                 this_entry := SYMTAB entry number of operand;
                 Generate IC '(IS, code)(S, this_entry)';
3. (Processing of END statement)
   (a) Perform step 2(b).
   (b) Generate IC '(AD,02)'.
   (c) Go to Pass II.
```

Scanned by CamScanner