Here is a method for running C routines as parallel processes. As you read the explanation keep in mind the basic idea (which is quite simple): ‘‘Execute one statement and then return, remembering where you were so that the next time you are called, you can execute the next statement’’. (Essentially, you are employing coroutine semantics; which C doesn’t have.) After this is done to each of say three procedures Z1, Z2, and Z3 just have a main program

```
loop
   Z1
   Z2
   Z3
end loop
```

The method is illustrated by the following example.

```
S2=0;
S=0;
for (j=0; j<n; j++) {
         S=S+X[j];
         if (X[j]>Q) {
              T=T+1;
              Z=2;
         } else {
              T=T-1;
              Z=1;
         }
    S2=S2+X[j]*X[j];
    }
S3=S+S2;
```

Step 1 Replace all loops by equivalent IF statements (plus gotos and assignment statements). This gives:

```
       S2=0;
       S=0;
       J=0;
   L001: ;
       if (j>=N) goto L002;
       S=S+X[j];
       if (X[j]>Q) {
          T=T+1;
          Z=2;
       } else {
          T=T-1;
          Z=1;
       }
       S2=S2+X[j]*X[j];
       J=J+1;
      goto L001
   L002: ;
       S3=S+S2;
```

Step 2: Between statements insert

PC = n; return; Ln: ;

(1) PC stands for program counter.

(2) n, initialized to 1, is a positive integer (a different one each time) Use n=1 before the first statement n=2 before the second, etc.

(3) The above doesn't work for IF-THEN-ELSE but a simple variant does (see example after step 3).

(4) You can eliminate labels added in step 1.

Step 3 At the begining of the program put

```
switch (PC) {
    case 1: goto L01;
    case 2: goto L02;
    case 3: goto L03;
    etc.
}
```

This gives for our example

```
switch (PC) {
case  1: goto L01;
case  2: goto L02;
case  3: goto L03;
...
case 13: goto L13;
}
                                                    L01: ;
S2=0;                           PC=2; return;   L02: ;
S=0;                            PC=3; return;   L03: ;
J=0;                            PC=4; return;   L04: ;
S=S+X[j];                       PC=5; return;   L05: ;

/* IF-THEN-ELSE   READ CAREFULLY */
if (X[j]>Q) PC=6; else PC=8;          return;   L06: ;
          T=T+1;                PC=7; return;   L07: ;
          Z=2;                  PC=10;return;   L08: ;
/* NOTICE JUMP IN PC - SKIPS OVER ELSE */
          T=T-1;                PC=9; return;   L09: ;
          Z=1;                  PC=10;return;   L10: ;
S2=S2+X[j]*X[j];                PC=11;return;   L11: ;
J=J+1;                          PC=12;return;   L12: ;

/* IF-THEN READ CAREFULLY */
if (j<N) PC=4; else PC=13;           return;   L13: ;
S3=S+S2;
```

Step 4 (optional) You can clean it up by putting the executable statements inside the case.

```
switch (PC) {

case  1: S2=0;              PC=2; return;
case  2: S=0;               PC=3; return;
case  3: J=0;               PC=4; return;
case  4: S=S+X[j];          PC=5; return;
case  5: if (X[j]>Q) PC=6; else PC=8; return;
       etc.
}
```