```
8.    lw R6,i
9.    add R7,R6,1
10.   sw R7,i
```

```
1.    li R1,save
2.    lw R2,i
3.    sll R3,R2,2
4.    add R4,R3,R1
5.    lw R5,0(R4)
6.    lw R6,k
7.    beq R5,R6,startwhileloop
```

FIGURE e2.15.4   A control flow graph for the *while* loop example. Each node represents a basic block, which terminates with a branch or by sequential fall-through into another basic block that is also the target of a branch. The IR statements have been numbered for ease in referring to them. The important transformation performed was to move the while test and conditional branch to the end. This eliminates the unconditional branch that was formerly inside the loop and places it before the loop. This transformation is so important that many compilers do it during the generation of the IR. The mult was also replaced with ("strength-reduced to") an sll.