1. In the IA-64’s (Itanium 2) *explicit parallel instruction computing (EPIC)* the compiler encodes multiple operations into a long instruction word so hardware can schedule these operations at run-time on multiple functional units without analysis, i.e., static multiple-issue. Why might the compiler be better able to find instructions that do not have dependencies than run-time hardware of a superscalar computer?

2. For a typical program on a traditional computer, more time is spent doing procedure/method calls than anything else. Why are procedure calls so time consuming?

3. What ways are parameters passed during a procedure call?

4. The Itanium assembly language can eliminate branch instructions by using predicate registers (e.g., p2, p3) as the following code:

   ```
   if (R1 == R2)
       R3 = R3 + R1
   else
       R3 = R3 - R1
   end if
   ```

   ```
   cmp.eq p2, p3 = r1, r2
   (p2) add r3 = r3, r1
   (p3) sub r3 = r3, r1
   ```

   However, why does this technique only improve performance if the “then” and “else” bodys are small sections of code?