To: X3J3
From: /HPC
Subject: Assertions

Goals:

1. Allow programmers to say things about their program that a compiler is unlikely or unable to determine on it's own.
2. Make explicit the difference between likelihood vs. *truth.*
3. Satisfy (at least partially) the request for "performance directives"

Non-goal: allow users to tell their processor things that are processor specific.

Illustrative syntax:

```
ASSERT  N > 64
DO I =1,N
      STUFF
END  DO
```

This would allow a processor to generate code that can 100% rely on N > 64. A processor *may* provide a mode of operation where there is a runtime check. Typically this will be used for more exciting expressions (M>N to assure no-overlap, mod(n,3)==0 to eliminate the need for "cleanup logic" when unrolling by a factor of 3 (and perhaps a hint to the optimizer that factors of 3 are a better choice than other factors, etc.)) This form of the assertion to apply to the next construct (in this case the DO LOOP) only.

In addition, a BLOCK … END BLOCK may be specified to allow more flexible control of the range of an assertion. An assertion applies to code in the current scope only.

For the *possibly* true syntax might look something like:

```
ASSERT (X)  logical_expression ! where x is a default kind real 0<= x <=1.0
```

0. means that the processor should assume that the probability of the expression is arbitrarily close to 0 (that is, can happen but almost never will). 1. Means that the processor should assume that the probability of the logical expression is arbitrarily close to certainty. A specific example:

```
ASSERT (.95) N > 64
DO I = 1, N
```

Would suggest that a processor with vector registers may assume that 95% of the time doing this computation in the vector units would be profitable. (.95) N < 5 would strongly suggest scalar processing would be best.