Fortran 90 and HPF

High-Performance Fortran

Reference Links

- http://www.nsc.liu.se/~boeln/f77to90/f77to90.html
- http://loki.stockton.edu/~stk7857/Fortran.htm
- http://www.utexas.edu/cc/parallel/HPF/
  (for pghpf compiler, available on turing; also see for more links)

Fortran

- "Formula Translation"
- A venerable language, used in (some) scientific computing circles
- One of the oldest surviving languages with the same name
- Invented by John Backus of IBM in 1956 (who since became a proponent of functional programming)
- Contemporary of Algol-60

Backus Quote

I don't know what the technical characteristics of the standard language for scientific and engineering computation in the year 2000 will be

... but I know it will be called Fortran.

John Backus, 1980's

Fortran Background

Original Fortran Manual
Typical Fortran Program

```fortran
program gauss
  c
  c       this program does a gauss-seidel iteration to solve a set of
  c       simultaneous equations with the coefficients a and rhs b
  double precision a(6, 6), x(6), b(6), sum, oldx, maxdiff, max_error
  integer i, j, n, steps
  max_error = 0.00001d0

  c       initial guess
  c
  do 100 i = 1, n
    x(i) = 0.0d0
  100     continue
  steps = 0
  150     continue
  maxdiff = 0.0d0
  do 300 i = 1, n
    oldx = x(i)
    sum = 0.0d0
    do 200 j = 1, n
      if( i .ne. j ) then
        sum = sum + a(i, j)*x(j)
      endif
    200         continue
    x(i) = (b(i)  - sum)/a(i, i)
    maxdiff = max(maxdiff, abs(x(i)-oldx))
  300         continue
  steps = steps + 1
  if( maxdiff .gt. max_error ) then
    go to 150
  endif
end
```

Fortran Coding Sheet
(historical artifact)

Fortran is Conservative

- It adds things like recursion, dynamic memory, and pointers 20 or more years after their appearance in other languages.

Fortran is Radical

- It adds things like array operations, array cross sections, distribution, and compiles them for parallel machines.
- It is one of the most optimizable and optimized languages.

Fortran is Surprising

- Given recent emphases on use for parallel computing, it is surprising that Fortran retains features that present obstacles:
  - Explicit ways to share (ALIAS) memory locations:
    - COMMON
    - EQUIVALENCE
Fortran 90

- Data parallel:
  - Entire arrays or sections of arrays can be operated on:
    - pairwise, or
    - by reduction operators
  - "where" construct for selective operations
  - "stride" for non-contiguous chunks of arrays (gather/scatter)

Vector Sectioning

- \(1, 7, 3, 2\) denotes a constant vector
- \(V(1, 7, 3, 2)\), where \(V\) is a vector, denotes the vector \(V(1), V(7), V(3), V(2)\)
- When a vector subscript is used on the LHS of an assignment \(V(1, 7, 3, 2) = W\) each index must be distinct.

Array Intrinsic Functions

- \(\text{maxval}(A)\)
- \(\text{maxloc}(A)\)
- \(\text{sum}(A)\)
- \(\text{dot_product}(A, B)\)
- \(\text{transpose}(A)\)
- \(\text{cyclic_shift}(A, \text{shift}, \text{dim})\)

HPF FORALL Statement

- Similar to DO statement, except
- Body can be executed in any order or in parallel
- Barrier between each statement in body

```fortran
REAL, DIMENSION(N, N) :: A, B
...
FORALL (I = 2:N-1, J = 2:N-1)
  A(I, J) = 0.25*(A(I, J-1)+A(I, J+1)+A(I-1,J)+A(I+1,J))
  B(I, J) = A(I, J)
END FORALL
```

FORALL Semantics

Nested FORALL Semantics
FORALL in Gaussian Elim.

HPF Alignment Directives

**REAL** A(1000), B(1000), C(1000), X(500), Y(0:500)
**INTEGER** INX(1000)

!HPF$ PROCESSORS PROC(10)

!HPF$ ALIGN X(I) WITH Y(I-1)

!HPF$ ALIGN X(I) WITH PROC(1+50)

!HPF$ ALIGN D(I, *) WITH PROC(I)

Virtual Processor Array

Processor Alignment

Array Alignment

The HPF Directives Allow Computation of Communication Costs

The HPF Directives Allow Computation of Communication Costs
**HPF Data Mapping**

![Diagram of HPF data mapping](image)

**“Independent” Directive**

- Specifies that loop bodies are to be regarded as executable in parallel.
- The compiler can be optimized based on this.
- Below it is possible that the same A(J) could be assigned to twice (non-deterministically).
- INDEPENDENT says it doesn’t matter

```
HPF5 INDEPENDENT
DO I = 1 TO N
   A(INDEX(I)) = B(I)
END DO
```

**“NEW” Directive**

- Prescribes variables for which new storage is allocated.

```
HPF5 INDEPENDENT, NEW(temp)
DO I = 1 TO N
   temp = A(I) + B(I)
   A(I) = temp
   B(I) = temp
END DO
```