If Conversion
Chapter 7, section 2

Vivek Pai
Comp 515
March 30, 1995

Problem

Control flow complicates dependence testing and vectorization

Goal

Convert control flow to conditional execution

Example

Before

```
DO I = 1, N
  IF (X(I) > 0) X(I) = X(I) + Y(I)
ENDDO
```

After

```
WHERE (X(1:N) > 0)
  X(1:N) = X(1:N) + Y(1:N)
```

General Idea

Determine what conditions are needed to execute

Condition execution on those conditions

Example

```
      DO 100 I = 1, N
S0      IF (X(I) < 0) GO TO 100
S1        X(I) = X(I) + Y(I)
100   CONTINUE
```

S1 executes only when S0 fails

Example

Introduce "guard" variable m(I)

```
      DO 100 I = 1, N
S0      m(I) = (X(I) < 0)
S1        if (!m(I))
              X(I) = X(I) + Y(I)
100   CONTINUE
```

Example

Vectorize

```
S0   m(1:N) = (X(1:N) < 0)
S1      where (!m(1:N))
           X(1:N) = X(1:N) + Y(1:N)
```

Simplest Case

Single loop

Only forward jumps

Why?
Things can only get skipped

Example

```
      DO 100 I = 1,N
C1      IF (A(I)>10) GO TO 60
20        A(I) = A(I) + 10
C2        IF (B(I)>10) GO TO 80
40          B(I) = B(I) + 10
60      A(I) = B(I) + A(I)
80      B(I) = A(I) - 5
      ENDDO
```

Example

```
      DO 100 I = 1,N
C1      IF (A(I)>10) GO TO 60
20        A(I) = A(I) + 10
C2        IF (B(I)>10) GO TO 80
40          B(I) = B(I) + 10
60      A(I) = B(I) + A(I)
80      B(I) = A(I) - 5
      ENDDO
```

C1's test must fail

Example

```
      DO 100 I = 1,N
C1      IF (A(I)>10) GO TO 60
20        A(I) = A(I) + 10
C2        IF (B(I)>10) GO TO 80
40          B(I) = B(I) + 10
60      A(I) = B(I) + A(I)
80      B(I) = A(I) - 5
      ENDDO
```

Executed if C1 is true

Example

```
      DO 100 I = 1,N
C1      IF (A(I)>10) GO TO 60
20        A(I) = A(I) + 10
C2        IF (B(I)>10) GO TO 80
40          B(I) = B(I) + 10
60      A(I) = B(I) + A(I)
80      B(I) = A(I) - 5
      ENDDO
```

Executed if C2 fails (and C1 fails)

Example

```
      DO 100 I = 1,N
C1      IF (A(I)>10) GO TO 60
20        A(I) = A(I) + 10
C2        IF (B(I)>10) GO TO 80
40          B(I) = B(I) + 10
60      A(I) = B(I) + A(I)
80      B(I) = A(I) - 5
      ENDDO
```

Executed if C2 succeeds

Example

```
      DO 100 I = 1,N
C1      IF (A(I)>10) GO TO 60
20        A(I) = A(I) + 10
C2        IF (B(I)>10) GO TO 80
40          B(I) = B(I) + 10
60      A(I) = B(I) + A(I)
80      B(I) = A(I) - 5
      ENDDO
```

Fall–through from 40

After If Conversion

```
      DO 100 I = 1,N
        m1(I) = A(I)>10

20      IF (!m1(I)) A(I) = A(I) + 10
        IF (!m1(I)) m2(I) = B(I)>10

40      IF (!m1(I) && !m2(I))
          B(I) = B(I) + 10

60      IF ((!m1(I) && !m2(I)) !! m1(I))
          A(I) = B(I) + A(I)

80      IF ((!m1(I) && !m2(I)) || m1(I) ||
             (!m1(I) && m2(I)))
          B(I) = A(I) - 5

      ENDDO
```

After Simplification

```
      DO 100 I = 1,N
        m1(I) = A(I)>10

20      IF (!m1(I)) A(I) = A(I) + 10
        IF (!m1(I)) m2(I) = B(I)>10

40      IF (!m1(I) && !m2(I))
          B(I) = B(I) + 10

60      IF (m1(I) || !m2(I))
          A(I) = B(I) + A(I)

80      B(I) = A(I) - 5

      ENDDO
```

Algorithm Sketch

Assume all statements can be executed
Keep a variable with 'current'
guard condition state

Walk statements of loop

'If (C) goto L' becomes
If (current) m = C
current = current && (!m)
cond(L) = cond(L) || (current && m)

'If (C) S' becomes
If (C && current) S

Otherwise, 'S' becomes
If (current) S

Algorithm

Loop Nests & Branches

Forward & Backward

Branch to same level in nest

Exit

Jumps out of current level

Removing Branches

Branch Relocation

Move exit branch to same level as target

Branch removal

Compute guard statements, conditionally execute everything

Exit branches

Condition statement execution on orginal taken/not taken behavior

Affects all statements in loop

```
      DO I =
        S1
        IF (P(I)) GO TO 100
        S2
      ENDDO

  100 S4
```

Example

```
DO J = 1,M
   DO I = 1,N
      A(I,J) = B(I,J) + X
      IF (!L(I,J)) GO TO 20
      C(I,J) = A(I,J) + Y
   ENDDO
   D(J) = A(N,J)
20 F(J) = C(10,J)
ENDDO
```

Introduce exit var.

```
DO J = 1,M
    lm = TRUE
    DO I = 1,N
        IF (lm) A(I,J) = B(I,J) + X
        IF (lm) lm = lm && L(I,J)
        IF (lm) C(I,J) = A(I,J) + Y
    ENDDO
    IF (!lm) GO TO 20
    D(J) = A(N,J)
20  F(J) = C(10,J)
ENDDO
```

If Conversion

```
DO J = 1,M
    lm = TRUE
    DO I = 1,N
        IF (lm) A(I,J) = B(I,J) + X
        IF (lm) lm = lm && L(I,J)
        IF (lm) C(I,J) = A(I,J) + Y
    ENDDO
    m = !lm
    IF (!m) D(J) = A(N,J)
20  F(J) = C(10,J)
ENDDO
```

Scalar Expansion

```
DO J = 1,M
    lm(0,J) = TRUE
    DO I = 1,N
        IF (lm(I-1,J))
            A(I,J) = B(I,J) + X
        IF (lm(I-1,J)) lm(I,J) =
            lm(I-1,J) && L(I,J)
        IF (lm(I,J))
            C(I,J) = A(I,J) + Y
    ENDDO
    IF (lm(N,J)) D(J) = A(N,J)
20 F(J) = C(10,J)
ENDDO
```

Vectorization

```
DO J = 1,M
    lm(0,J) = TRUE
    DO I = 1,N
        IF (lm(I-1,J)) lm(I,J) =
            lm(I-1,J) && L(I,J)
    ENDDO
ENDDO

WHERE (lm(0:N-1,1:M))
    A(1:N,1:M) = B(1:N,1:M) + X
WHERE (lm(0:N-1,1:M))
    C(1:N,1:M) = A(1:N,1:M) + Y
WHERE (lm(N,1:M)) D(1:M) = A(N,1:M)
20 F(1:M) = C(10,1:M)
```

Alternative

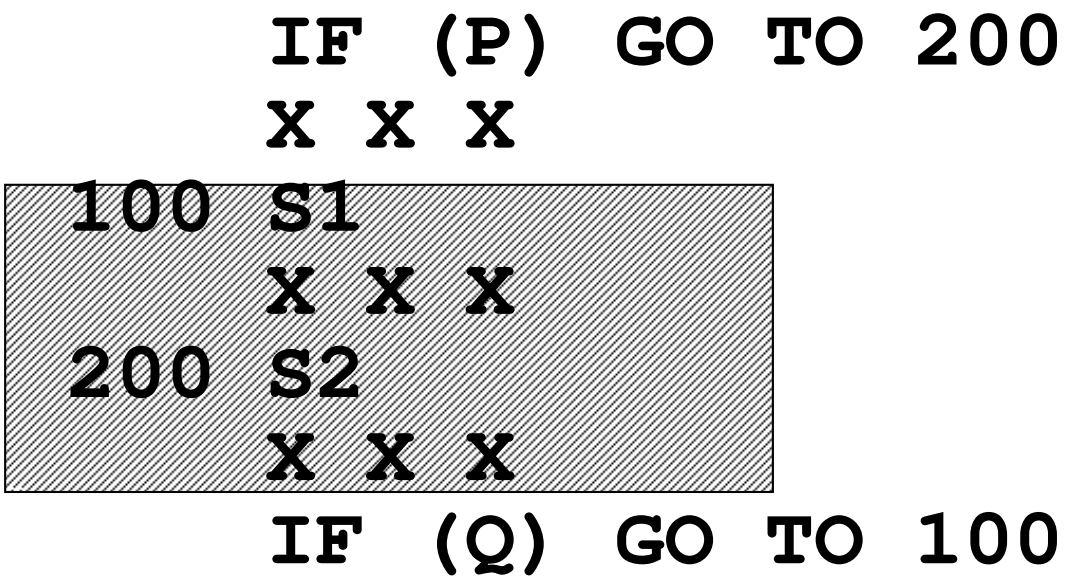
```
DO J = 1,M
   lm(0) = TRUE
   DO I = 1,N
      IF (lm(I-1)) lm(I) =
         lm(I-1) && L(I,J)
   ENDDO

   WHERE (lm(0:N-1))
      A(1:N,J) = B(1:N,J) + X
   WHERE (lm(1:N))
      C(1:N,J) = A(1:N,J) + Y
   IF (lm(N)) D(J) = A(N,J)
   20 F(J) = C(10,J)
ENDDO
```

Improved

```
DO J = 1,M
   n = FirstFalseX(L(1:N,J))
   a = min(n,N)
   A(1:a,J) = B(1:a,J) + X
   C(1:n-1,J) = A(1:n-1,J) + Y
   IF (n > N) D(J) = A(N,J)
   20 F(J) = C(10,J)
ENDDO
```

Algorithm

Backward Branches



‘Implicity Iterative Region’

Two Possibilities

S1 executed on first pass (P fails)

S1 executed on backward branch

Transformed

```
      m = P
      X X X
      bb = FALSE
100   IF (!m || (m&&bb)) S1
      <guard> X X X
200   <guard> S2
      <guard> X X X
      IF (Q) THEN
        bb = TRUE
        GO TO 100
      ENDIF
```

DO Loops

```
 20 DO 100 I = 1,100
 40    L = 2*I
 60    DO 100 J = 1,L
 80       A(I,J) = 0
100 CONTINUE
```

Iteration counts

```
      ICNT1 = 100
 40     L = 2*I (ICNT1)
 60     ICNT2 = L
 80       A(I,J) = 0 (ICNT2)
100 CONTINUE
```

```
 20 DO 100 I = 1, 100
 40     L = 2*I
 80     A(I,1:L) = 0
100 CONTINUE
```

Boolean simplification

Fast, simple algorithm

We create conditions in a certain order

We can use that info to simplify

Simplify expressions from left