affected by the vector instruction.

- If \( \text{VM}(i) = 0 \) the \( i \)-th component of the result is not

- If \( \text{VM}(i) = 1 \) the \( i \)-th component of the result is the \( i \)-th

component produced by the vector operation.

Definition: A vector mask \( \text{VM} \) is a bit vector which controls

\textbf{Vector Mask}
end do
end if

\( C(I) = A(I) + B(I) \)

if \( WM(I) = I \) then

\( d(I) = 1 \), \( N \)

\( WM \) is

Example: \( C[I:N:1] = [A[I:N:1] + B[I:N:1]] \) controlled by
Such knowledge is typically important from an efficiency standpoint.

- Vectors.
  
  and deduce. Sometimes it is only true on segments of running a few tests with carefully chosen kernels to time sure if this information is available. If not it is worth

- When writing in terms of masks always check to make
  
  pipeline but it need not always be.

- This is usually the case when a mask is controlling a

  the number of ones in WM.

- The number of operations performed is equal to N not
Definition: COMPRESS_V0_V1 is defined to be

Instructions

The mask allows the definition of other data handling
Definition: EXPAND V0 VT is defined to be

end do

end if

pfr = pfr + 1

V0(1) = (V1(1))

if V1(1) = 1 then

do if VT = 1, V1

pfr = 1

if
architecture definition if it becomes crucial to efficiency.

Be careful what you assume about the X's. Check the

\[
V_I (X) = V_M (0, 0, 0, 1) = V_0 (1, 2, 3, 0)
\]

EXPAND V0 INTO V1 USING VM

\[
V_I (X) = V_M (0, 0, 0, 1) = V_0 (1, 2, 3, 0)
\]

COMPRESS V0 INTO V1 USING VM
end do

VZ(1)=MEM(V1(1))
do 1 = 1, V1

MEM, V1, V2 is defined to be

Definition: A GATHER operation, GATHER

Other special data handling instructions are also possible
end do

\( \text{MEM}(V_1(\tau)) = V_2(\tau) \)

do \( i = 1, V_L \)

MEM, \( V_1, V_2 \) is defined to be

**Definition:** A SCATTER operation, SCATTER
\[ V_2 = (20', 40', 30', 50', 10') \]
\[ V_1 = (2', 4', 3', 5', 1') \]
\[ WEM = (10', 20', 30', 40', 50') \]

GATHER WEM INTO V2 USING V1
specified.

If you are not using a permutation, correctness must be

\[ \text{MEM} = (10, 20, 30, 40, 50) \]

so after SCATTER independently of initial values.

scatter inverts this operation.
independent of the array \( y \).

where \( f(x(t)) \) is some vectorizable function that is

\[
\begin{align*}
\text{end do} \\
((\tau)x)\downarrow = (\tau)\downarrow \quad ((\tau)\downarrow)^t \quad \text{do} \quad \downarrow = \downarrow, \quad \text{u}
\end{align*}
\]

Consider the following Fortran loop:

Vector Conditionals
handling instructions (compress/expand or scatter/gather).

Implement the code above without using and special data.

mentioned above.

vector instructions as does the value of the VTR register.
the length, in elements, of each vector register (controls all
Recall that the vector mask register (R-bits long where R is
end do

SY A2',V3
AND VM UNCHANGED

WITH THE RESULT INTO V3

UNDER CONTROL OF VM

CODE HERE TO IMPLEMENT P(X)

L>Y V1',V2
LV V2',A2
LV V1',A1

LEA A2',Y(1+(1-I)+R)
LEA A1',X(1+(1-I)+R)

CM

DO I=1,K

MOVSTL VLR'R
total number of operations is proportional to $n$

full vector length computations $\rightarrow$ good performance
number of trues in the entire set of n tests.

trues generated in each block of R elements not by the total

Note that the vector lengths are determined by the number of

memory.

If is costly, e.g., there is no point if it is just a vector store to

that satisfies the logical condition. This only makes sense if

the function \( R \) and therefore only operate on the elements

Compress and expand can be used to compress the input to
MOVS2S RI, VM
SAVE VM IN A REGISTER **

L>V V1, V2
L V2, A2
L V1, A1

LEA A2, Y(I+1)\*R)
LEA A1, X(I+1)\*R)

CM

do I=1, K

MOVS2L VLR, R
end do

Sad V3

EXPAND V3, V4

MOVSL2 VM, R1

MOVSL VLR, R

RESTORE OLD VLR AND VM
WITH THE RESULT INTO V4 AND WM UNCHANGED
WITH SHORTENED VLR AND Full WM
(CODE HERE TO IMPLEMENT F(X))

CM

MOVSL2 VLR, R2

POP R2, VM

COMPRSS VS3, V1

COMPRRESS INPT TO F(X)

**
scattered into \( \times \) using the same index vector.

evaluate \( R \) with full vector length. The results are then
appropriate elements of the vector \( X \) in groups of \( R \) and to
an index vector that can be used to gather all of the
condition evaluates true. The set of such indices determines
determining the indices of the elements for which the logical
One can remove this problem of local true vector lengths by
vector of at most \( R \) is needed.

complex control for more efficient use of space (can index computed and the results scattered. This trades slightly more associated elements from \( X \) are then gathered and \( \mathbf{P} \) exactly \( R \) trues are found along with their indices. The need to produce a single index vector from all trues can be mitigated somewhat by performing the logical tests until
use of the mask shown in the first version. evaluate the false and suppress their stores with the simple work and just do the small amount of work needed to high, i.e., it would have been better to avoid the extra control. All of these approaches suffer if the density of trues is very