Vector Processor : Handling Loops/ Conditional Statements

1. Vector-Length Registers: Handling Loops Not Equal to 64 :

- A vector register processor has a natural vector length determined by the number of elements in each vector register. This length, which is 64 for VMIPS, is unlikely to match the real vector length in a program. Moreover, in a real program the length of a particular vector operation is often unknown at compile time. In fact, a single piece of code may require different vector lengths. For example, consider this code:

  ```
  For (i=0; i<n;   i=i+1)
  Y[i]= a*X[i]+ Y[i];
  ```

- The size of all the vector operations depends on n, which may not even be known until run time! The value of n might also be a parameter to a procedure containing the above loop and therefore subject to change during execution
- a vector-length register (VLR). The VLR controls the length of any vector operation, including a vector load or store. The value in the VLR, however, cannot be greater than the length of the vector registers. This solves our problem as long as the real length is less than or equal to the maximum vector length (MVL). The MVL determines the number of data elements in a vector of an architecture.

- What if the value of n is not known at compile time and thus may be greater than the MVL? To tackle the second problem where the vector is longer than the maximum length, a technique called strip mining is used. Strip mining is the generation of code such that each vector operation is done for a size less than or equal to the MVL. We create one loop that handles any number of iterations that is a multiple of the MVL and another loop that handles any remaining iterations and must be less than the MVL.

```
low=0;
  VL=(n% MVL);    /*find odd-size piece using modulo op%*/
```

```
for(j=0; j<_(n/MVL);   j=j+1)          { /*outer loop*/

 for (i= low; i< (low+VL);  i=i+1) /*runs for length VL*/
     Y[1]= a* X[i]+ Y[i];   /*main operation*/
      low= low+ VL;    /*start of next vector*/
      VL = MVL;   /*reset the length to maximum vector length*/
                  }
```

- The term n/MVL represents truncating integer division. The effect of this loop is to block the vector into segments that are then processed by the inner loop. The length of the first segment is (n % MVL), and all subsequent segments are of length MVL.

2. Vector Mask Registers: Handling IF Statements in Vector Loops :

- The presence of conditionals (IF statements) inside loops and the use of sparse matrices are two main reasons for lower levels of vectorization. Programs that contain IF statements in loops cannot be run in vector mode using the techniques we have discussed so far because the IF statements introduce control dependences into a loop.

- Consider the following loop written in C:

- ```
  For(I =0; i< 64;   i=i+1)
    if(X[i]!=0)
      X[i]= X[i]- Y[i];
  ```

- This loop cannot normally be vectorized because of the conditional execution of the body; however, if the inner loop could be run for the iterations for which X [i] # 0, then the subtraction could be vectorized.

- The common extension for this capability is vector-mask control. Mask registers essentially provide conditional execution of each element operation in avector instruction. The vector-mask control uses a Boolean vector to control the execution of a vector instruction, just as conditionally executed instructions use a Boolean condition to determine whether to execute a scalar instruction. When the vector mask register is enabled, any vector instructions executed operate only on the vector elements whose corresponding entries in the vector-mask register are one. The entries in the destination vector register that correspond to a zero in
  ```

the mask register are unaffected by the vector operation. Clearing the vector-mask register sets it to all ones, making subsequent vector instructions operate on all vector elements.

-

```
LV          V1,Rx      ;load vector X into V1
LV          V2,Ry      ;load vector Y
L.D         FO,#O      ;load FP zero into FO
SNEVS.D     V1,FO      ;sets VM(i) to1if V1(i)!=F0
SLIBVV.D    V1,V1,V2   ;subtract under vector mask
SV          V1,Rx      ;store the result in X
```

- Using a vector-mask register does have overhead, however. With scalar architectures, conditionally executed instructions still require execution time when the condition is not satisfied. Nonetheless, the elimination of a branch and the associated control dependences can make a conditional instruction faster even if it sometimes does useless work. Similarly, vector instructions executed with a vector mask still take the same execution time, even for the elements where the mask is zero.