How Vector Processors Work: An Example

- We can best understand a vector processor by looking at a vector loop for VMIPS. Let's take a typical vector problem, which we use throughout this section: Y = a* X+ Y

- Where X and Y are vectors, initially resident in memory, and a is a scalar. This problem is the so-called SAXPY or DAXPY loop that forms the inner loop of the Linpack benchmark. (SAXPY stands for single-precision a * X plus Y; DAXPY for double precision a * X plus Y.) Linpack is a collection of linear algebra routines, and the Linpack benchmark consists of routines for performing Gaussian elimination. For now, let us assume that the number of elements, or length, of a vector register (64) matches the length of the vector operation we are interested in.

Example : Show the code for MIPS and VMIPS for the DAXPY loop. Assume that the starting addresses of X and Y are in Rx and Ry, respectively.

Answer: Here is the MIPS code.'

```
      L.D       FO,a          ;load scalar a
      DADDIU    R4,Rx,#512    ;last address to load
Loop: L.D       F2,0(Rx)      ;load X[i]
      MUL.D     F2,F2,FO      ;a x X[i]
      L.D       F4,0(Ry)      ;load Y[i]
      ADD.D     F4,F4,F2      ;a x X[i]    + Y[i]
      S.D       F4,9(Ry)      ;store into Y[i]
      DADDIU    Rx,Rx,#8      ;increment index to X
      DADDIU    Ry,Ry,#8      ;increment index to Y
      DSUBU     R20,R4,Rx     ;compute bound
      BNEZ      R20,Loop      ;check if done
```

Here is the VMIPS code for DAXPY.

```
L.D        FO,a          ;load scalar a
LV         V1,Rx         ;load vector X
MULVS.D    V2,V1,F0      ;vector-scalar multiply
LV         V3,Ry         ;load vector Y
ADDVV.D    V4,V2,V3      ;add
SV         V4,Ry         ;store the result
```

- The most dramatic difference is that the vector processor greatly reduces the dynamic instruction bandwidth, executing only 6 instructions versus almost 600 for MIPS. This reduction occurs because the vector operations work on 64 elements and the overhead instructions that constitute nearly half the loop on MIPS are not present in the VMIPS code. When the compiler produces vector instructions for such a sequence and the resulting code spends much of its time running in vector mode, the code is said to be vectorized or vectorizable. Loops can be vectorized when they do not have dependences between iterations of a loop, which are called loop-carried dependences.

- Another important difference between MIPS and VMIPS is the frequency of pipeline interlocks. In the straightforward MIPS code, every ADD. D must wait for a MUL. D, and every S. D must wait for the ADD. D. On the vector processor, each vector instruction will only stall for the first element in each vector, and then subsequent elements will flow smoothly down the pipeline. Thus, pipeline stalls are required only once per vector instruction, rather than once per vector element. Vector architects call forwarding of element-dependent operations chaining, in that the dependent operations are "chained" together.