Vector Architecture

Vector architecture includes instruction set extensions to an ISA to support vector operations, which are deeply pipelined.

Vector operations are on vector registers, which are xed-length bank of registers. Data is transferred between a vector register and the memory system.

Each vector operation takes vector registers or a vector register and a scalar value as input.

Vector architecture can only be effective on applications that have significant data-level parallelism (DLP). Vector processing advantages greatly reduces the dynamic instruction bandwidth. Generally execution time is reduced due to

(1) Eliminating loop overhead

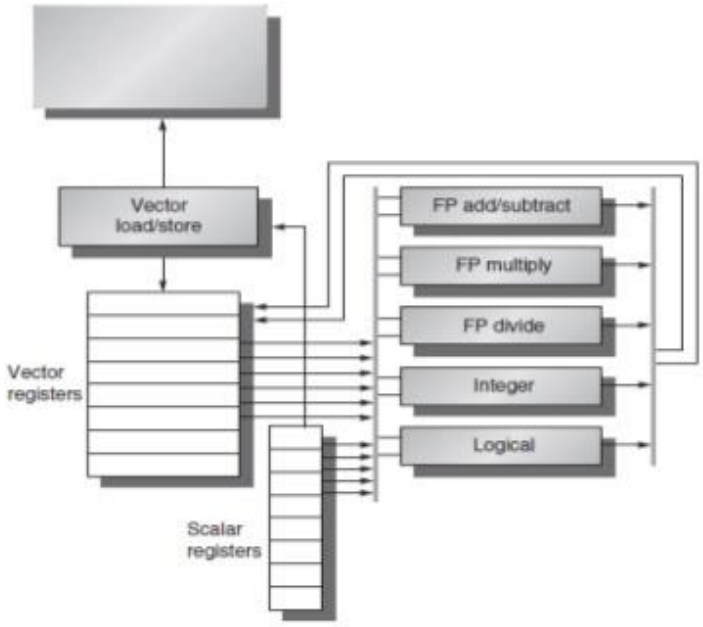
(2) Stalls only occurring on the first- vector element rather than on each vector element, and

(3) Performing vector operations in parallel.

Example of Vector Code

```
/* Scalar MIPS Code */          /* Source Code */
   L.D    F0,a          <=     for (i = 0; i < 64; i++)
   DADDIU R4,Rx,#512               Y[i] = a * X[i] + Y[i];
Loop:
   L.D    F2,0(Rx)
   MUL.D  F2,F2,F0              /* VMIPS Code */
   L.D    F4,0(Ry)              L.D     F0,a
   ADD.D  F4,F4,F2              LV      V1,Rx
   S.D    F4,0(Ry)       =>     MULVS.D V2,V1,F0
   DADDIU Rx,Rx,#8              LV      V3,Ry
   DADDIU Ry,Ry,#8              ADDVV.D V4,V2,V3
   DSUBU  R20,R4,Rx             SV      V4,Ry
   BNEZ   R20,Loop
```

Basic structure of a vector architecture



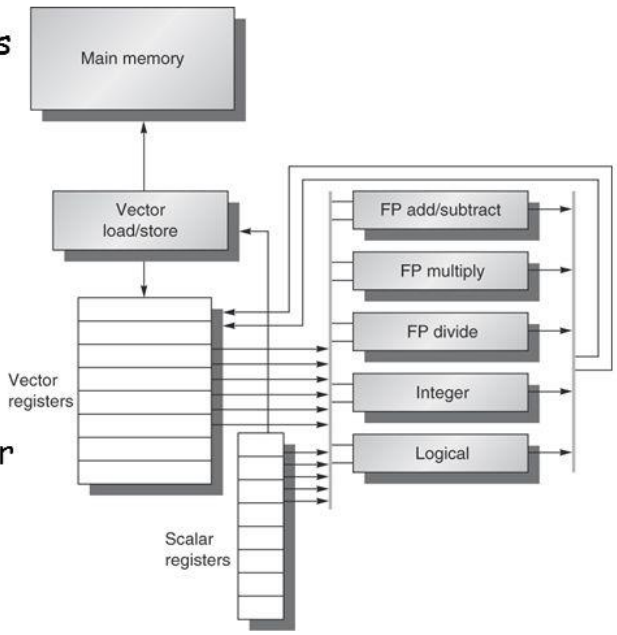
Vector Architecture

Basic idea:

- Read sets of data elements into "vector registers"
- Operate on those registers
- Disperse the results back into memory

Registers are controlled by compiler

- Used to hide memory latency
  - by loading data early (many cycles before their use)
- Leverage memory bandwidth



Components of vector architecture

- Vector registers
  - Each register holds a 64-element, 64 bits/element vector
  - Register file has 16 read ports and 8 write ports
- Vector functional units
  - Fully pipelined
  - Data and control hazards are detected
- Vector load-store unit
  - Fully pipelined
  - Words move between registers
  - One word per clock cycle after initial latency
- Scalar registers
  - 32 general-purpose registers
  - 32 floating-point registers