Vector Processors

A vector computer is a SIMD machine

Single vector instructions operate on multi-element data items

Commands are broadcast to the processing elements (PE)

PE's operate in lock-step fashion performing the same operation

Example: Z = s*X + Y where Z, X and Y are arrays or vectors
s is a scalar constant or variable

Intel-based PC's have MMX, SSE & AVX instructions

Multi-media extensions (integer)

Streaming SIMD extensions (floating point)

Advanced Vector Extensions (AVX)

- Operates on four 64-bit floating point numbers in parallel

Vector Processors

Only supercomputers had vector instructions in the past

the Cray-1 designed by Seymour Cray in the 70s

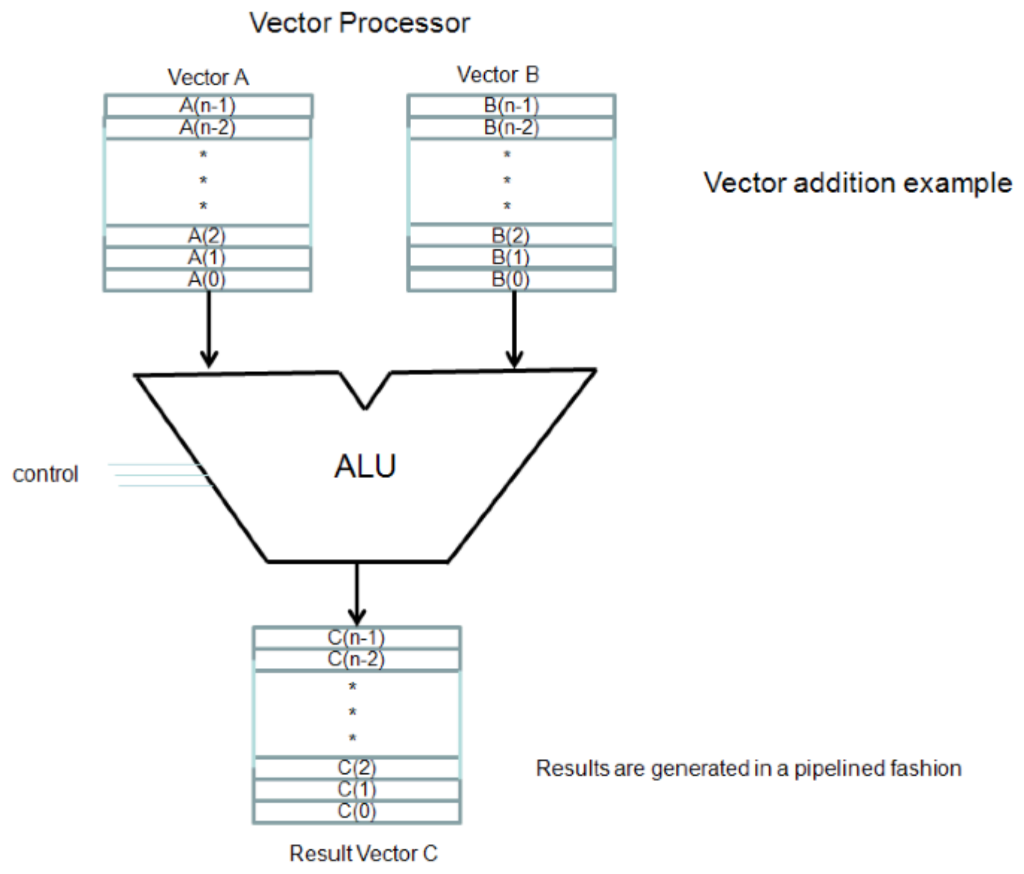
- Highly pipelined functional units
- Data get streamed to and from vector registers
  - Data collected from memory into registers
  - Results stored from registers to memory

Basic idea:

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

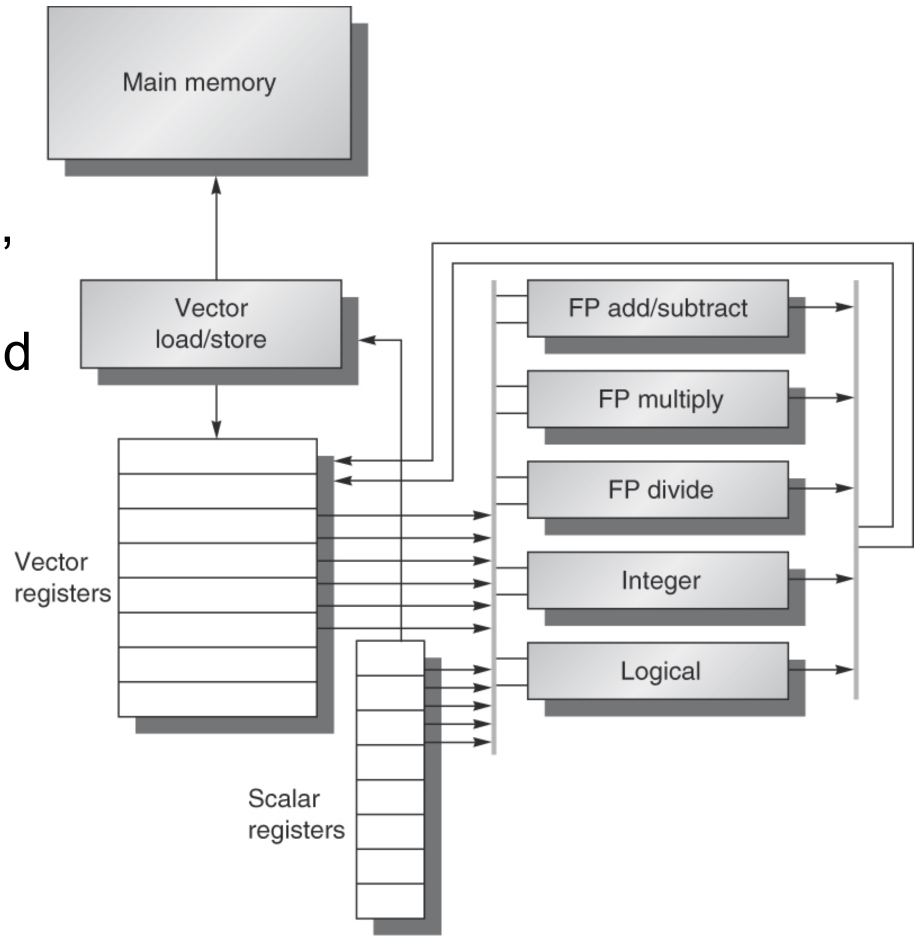
Vector Processors

High memory bandwidth is required to quickly fill registers



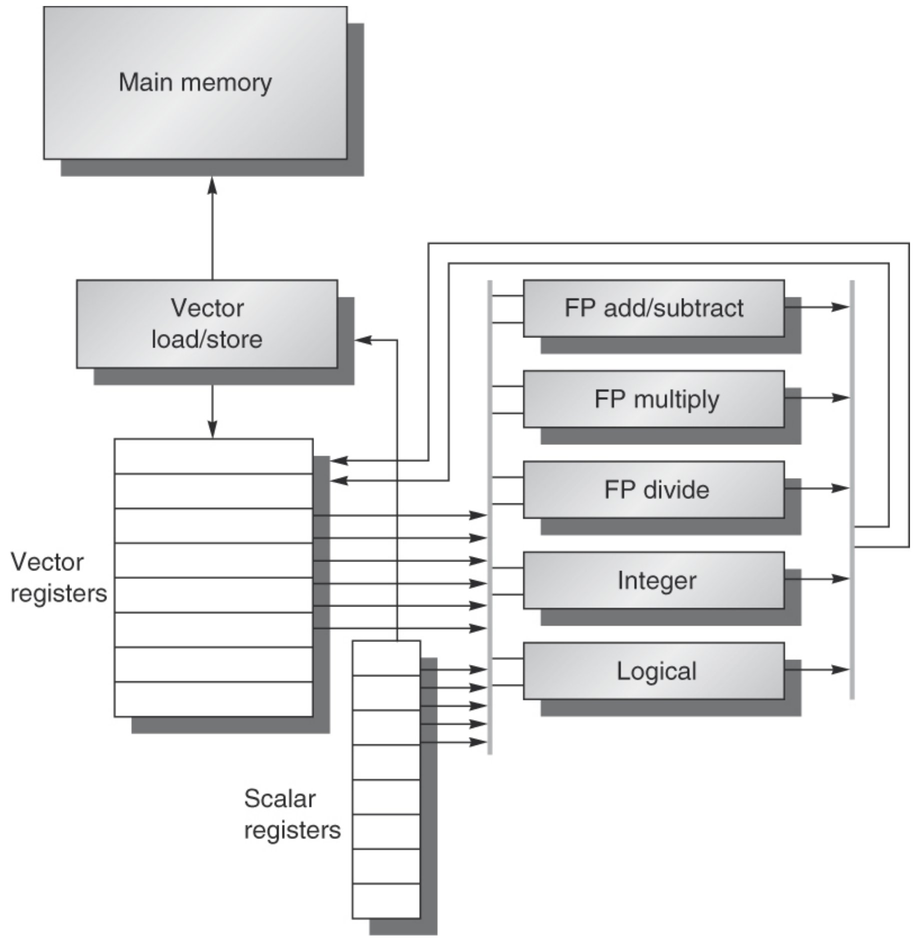
Example VMIPS

- Vector registers
  - Each register holds a 64-element, 64 bits/element vector
  - Register file has 16 read ports and 8 write ports



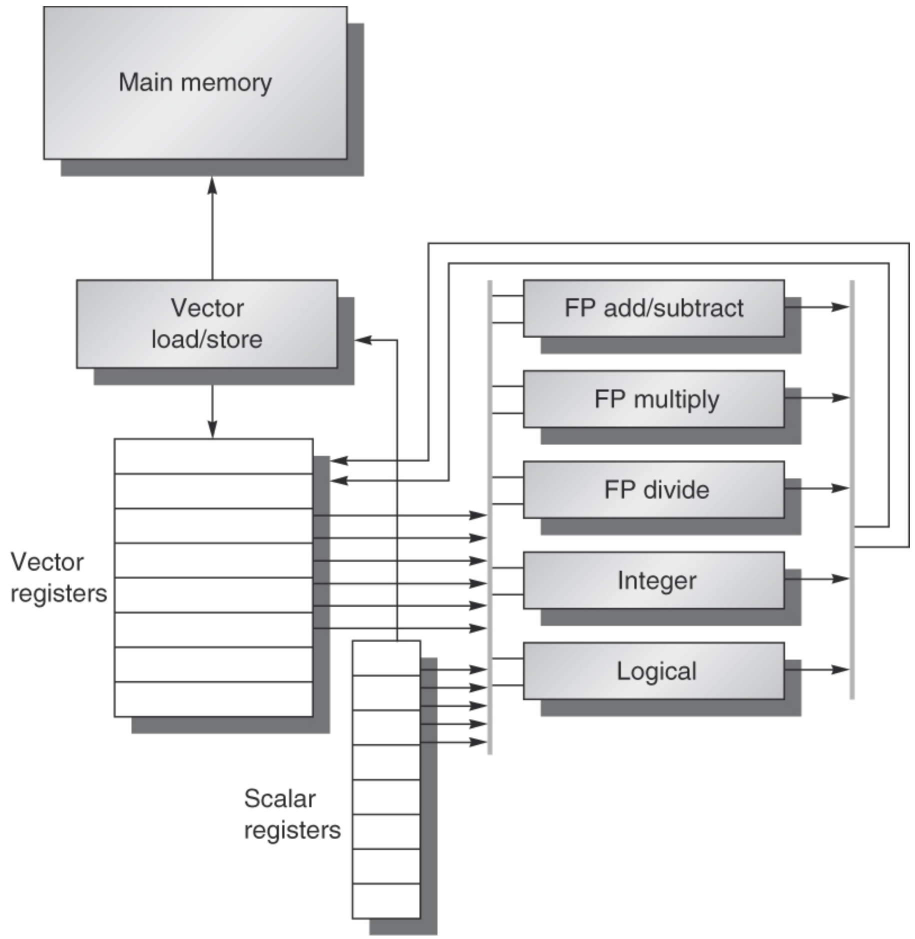
Example VMIPS

- Vector functional units
  - Fully pipelined
  - Data and control hazards are detected



Example VMIPS

- 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



Example VMIPS Instructions

```
# C code
for (i=0; i<64; i++)
 C[i] = A[i] + B[i];
```

```
# Scalar Code
  LI     R4, 64
loop:
  L.D    F0, 0(R1)
  L.D    F2, 0(R2)
  ADD.D  F4, F2, F0
  S.D    F4, 0(R3)
  DADDIU R1, 8
  DADDIU R2, 8
  DADDIU R3, 8
  DSUBIU R4, 1
  BNEZ   R4, loop
```

```
# Vector Code
  LI      VLR, 64
  LV      V1, R1
  LV      V2, R2
  ADDV.D  V3, V1, V2
  SV      V3, R3
```

Example VMIPS Instructions

- Example: DAXPY

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

- In MIPS Code
  - ADD waits for MUL, SD waits for ADD

- In VMIPS
  - Stall once for the first vector element, subsequent elements will flow smoothly down the pipeline.
  - Pipeline stall required once per vector instruction!

Example VMIPS Instructions

- Execution time depends on three factors:
  - Length of operand vectors
  - Structural hazards
  - Data dependencies
- VMIPS functional units consume one element per clock cycle
  - Execution time is approximately the vector length