Concurrent Processor

- **Vector processor**
- **Multiple Instruction Issue Processors**

Vector processor

In a vector processor, a single instruction operates simultaneously on multiple data items. (called "SIMD" in Flynn's taxonomy)

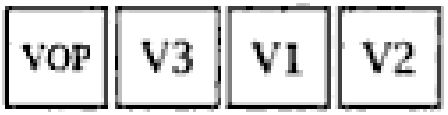
Terminology:

- Scalar – a single quantity (number).
- Vector – an ordered series of scalar quantities – a one-dimensional array.

Vector processor

$$\vec{A} + \vec{B} = \vec{C}$$

are represented by a vector instruction of the type



where VOP represents a vector operation, and V1, V2, V3 indicate specific vector registers. The operation performed is:

$$V3 := V1 \text{ VOP } V2$$

for all register values within each vector register.

The individual element within a vector register is designated V1.X. Thus, the vector elements of a 64-element vector register V1 are indicated as V1.1 through V1.64.

Vector processor

Vector Instructions are effective in several way:

1. **They significantly improve code density.**
2. **They reduce the number of instruction required to execute a program.**
3. **They organize the data argument into regular sequences that can be efficiently handled by the hardware.**
4. **They can be represent a simple loop construct, thus removing the control overhead for loop execution.**

Vector processor

Different accessing patterns:

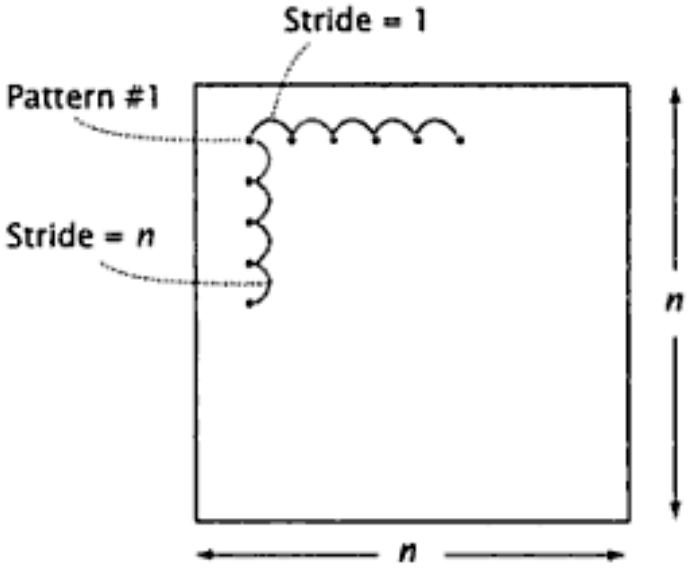


Figure 7.1 For an array in memory, different accessing patterns use different strides in accessing memory.

Accessing array element, separated by an addressing distance called the stride.

Vector processor

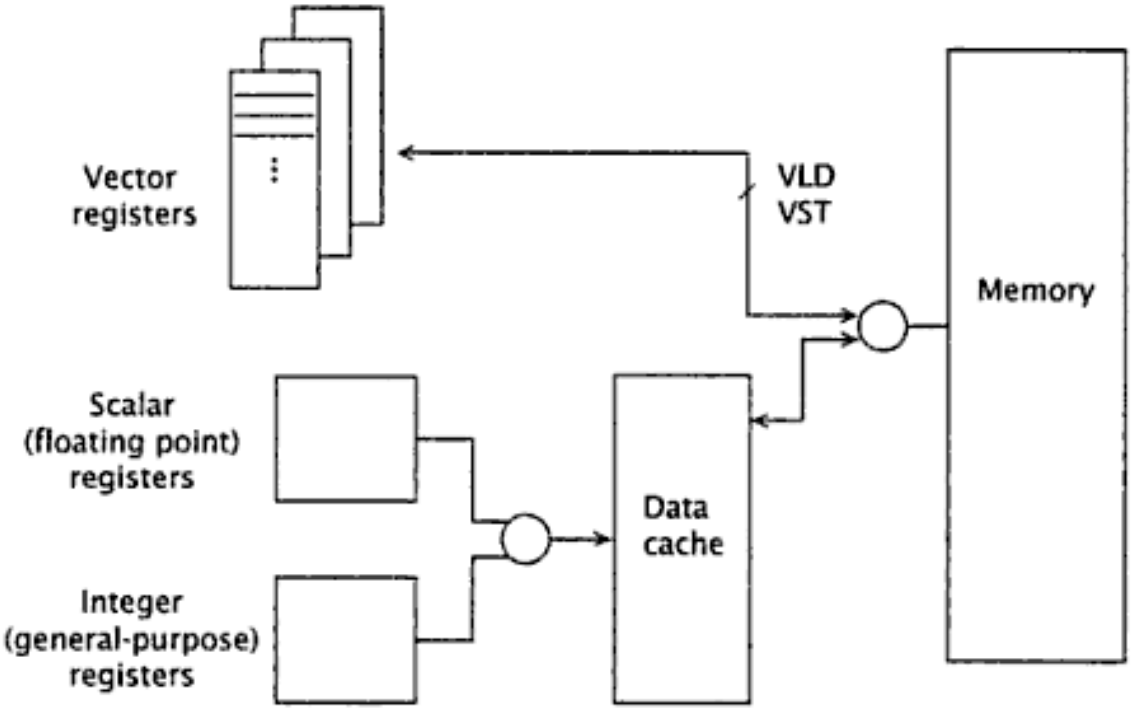


Figure 7.2 The primary storage facilities in a vector processor. Vector LD/ST usually bypasses the data cache.

Vector processor

Vector functional Unit:

The vector register consist of eight or more register sets, each consisting of 16-64 vector element. Each vector element is a floating-point word.

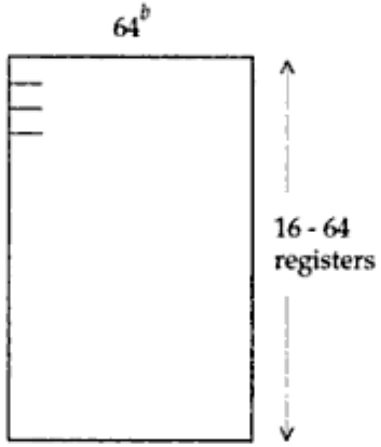


Figure 7.3 Typical vector register sizes.

Vector processor

Functional Unit for floating point:

1. **Add/ Subtract**
2. **Multiplication**
3. **Division**
4. **Logical Operation**

There are separate and independent functional unit to manage the load/ store function.

Following figure show the approximate timing for a sample 4-stage functional pipeline.

Vector processor

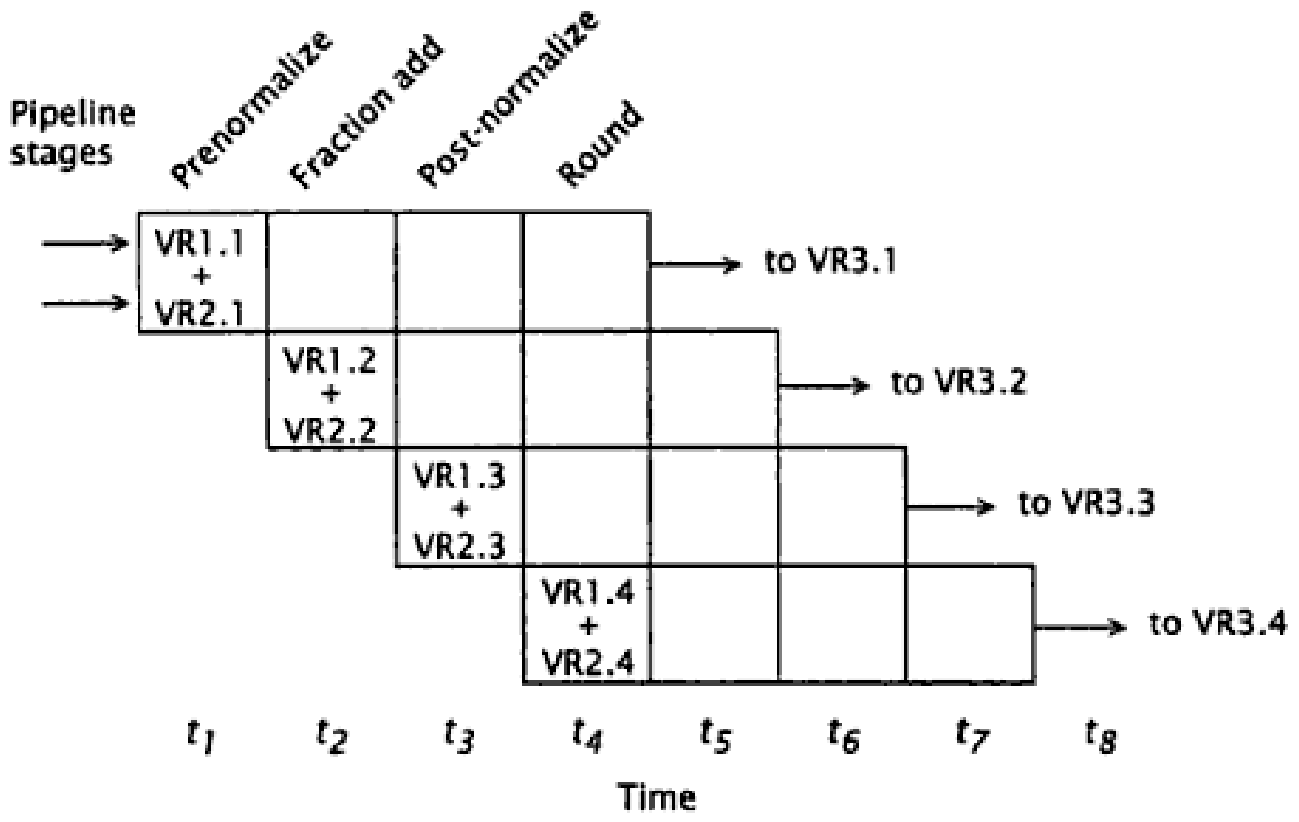


Figure 7.4 Approximate timing for a sample 4-stage functional pipeline.

Vector processor

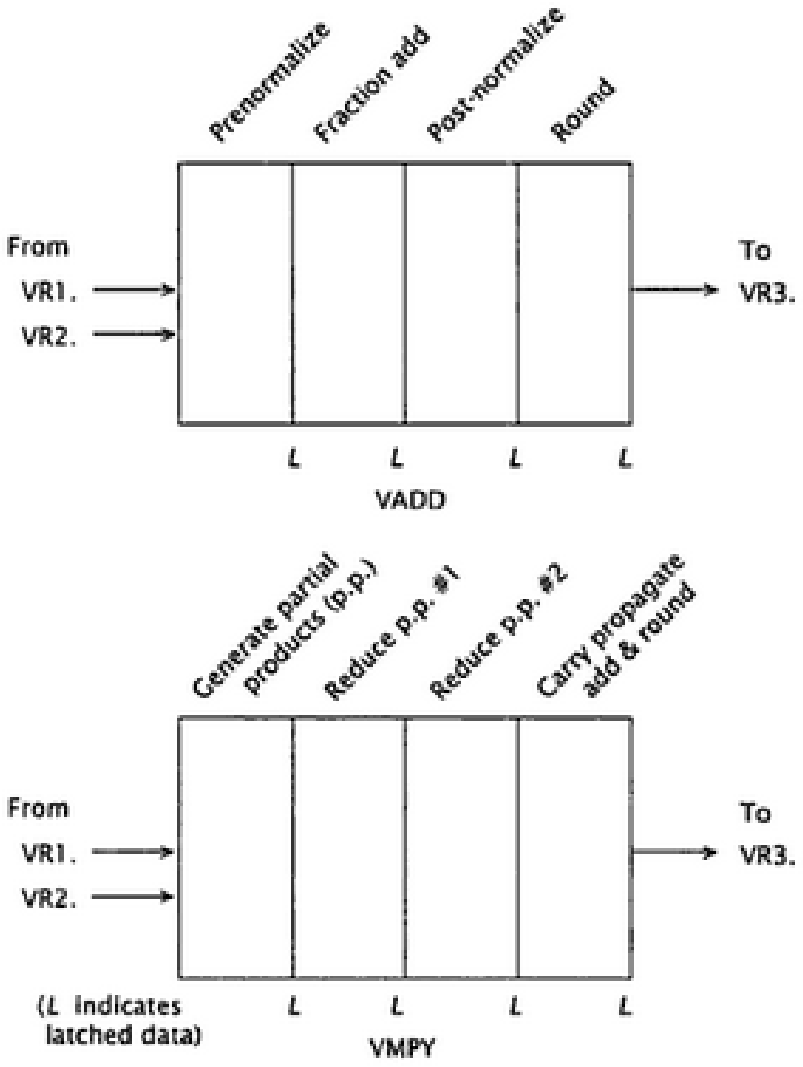


Figure 7.5 Two vector arithmetic units, each partitioned into four pipeline stages. Results are latched at each latch point (*L*).

Vector processor

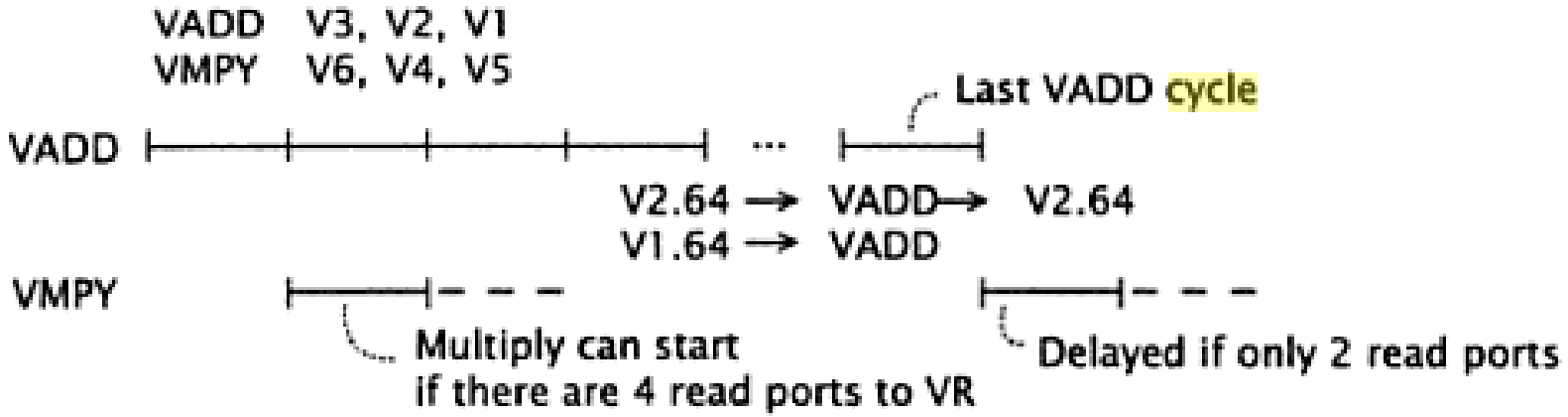


Figure 7.6 For logically independent vector instructions, the number of access paths to the vector register set and vector units may limit performance. If there are four read ports, the VMPY can start on the second cycle. Otherwise, with two ports, the VMPY must wait until the VADD completes use of the read ports.

Vector processor

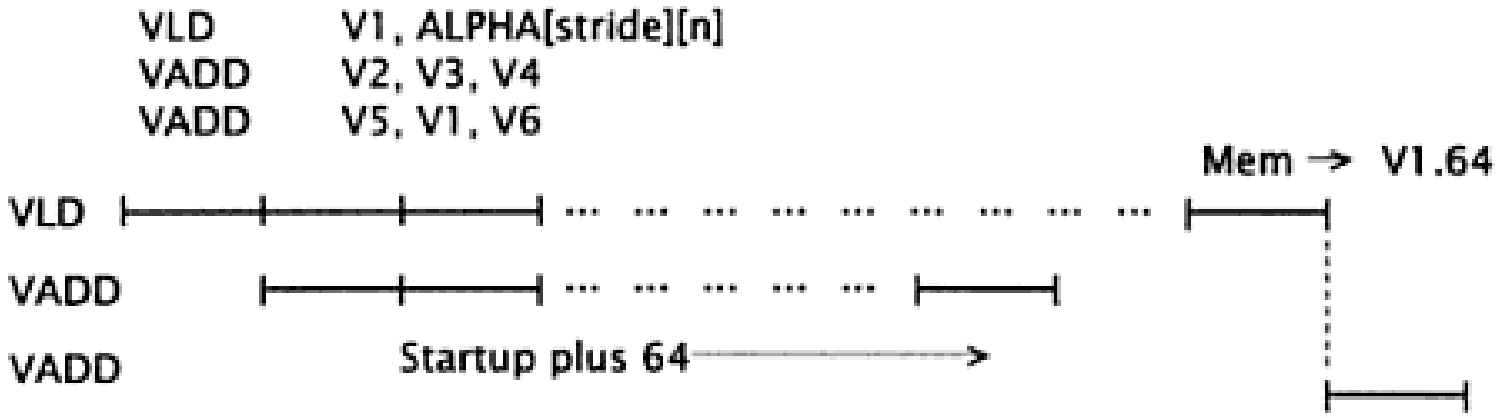
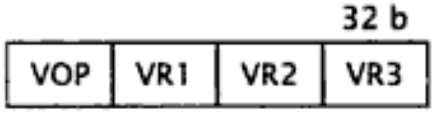


Figure 7.7 While independent VLD and VADD may proceed concurrently (with sufficient VR ports), operations that use the results of VLD do not begin until the VLD is fully complete.

Vector processor

Vector Instruction / operations



Vector instruction layout

(a) Vector operation: VR OP VR → VR

VADD
VSUB
VMPY
VDIV
VAND
VOR
VEOR

(b) Compare (VCOMP): VR VCOMP VR → Scalar

V1 VCOMP V2 → S

$S(i)$ bit is "1" if V1.i ≥ V2.i

Test (VTEST): V1 VTEST CC → Scalar

$S(i)$ bit is "1" if V1.i satisfies CC

CC is the condition code specified in the instruction

(c) Accumulate (VACC): VACC: $\sum(V1 * V2) \rightarrow S1$

VR VACC VR → Scalar

Vector processor

Vector chaining Path

For these two instructions:

VADD VR3, VR1, VR2

VMPY VR5, VR3, VR4

the timing would be:

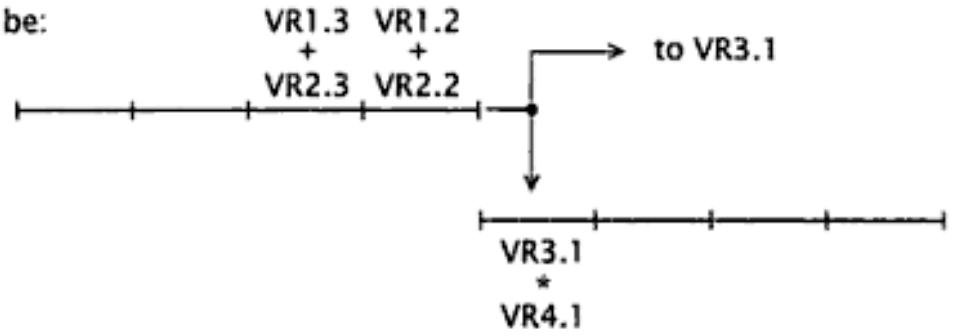


Figure 7.12 Effect of vector chaining.

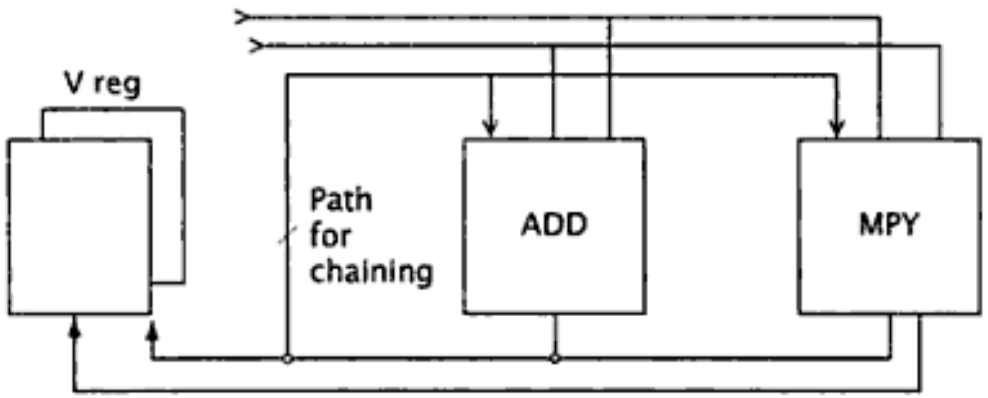


Figure 7.13 Vector chaining path.