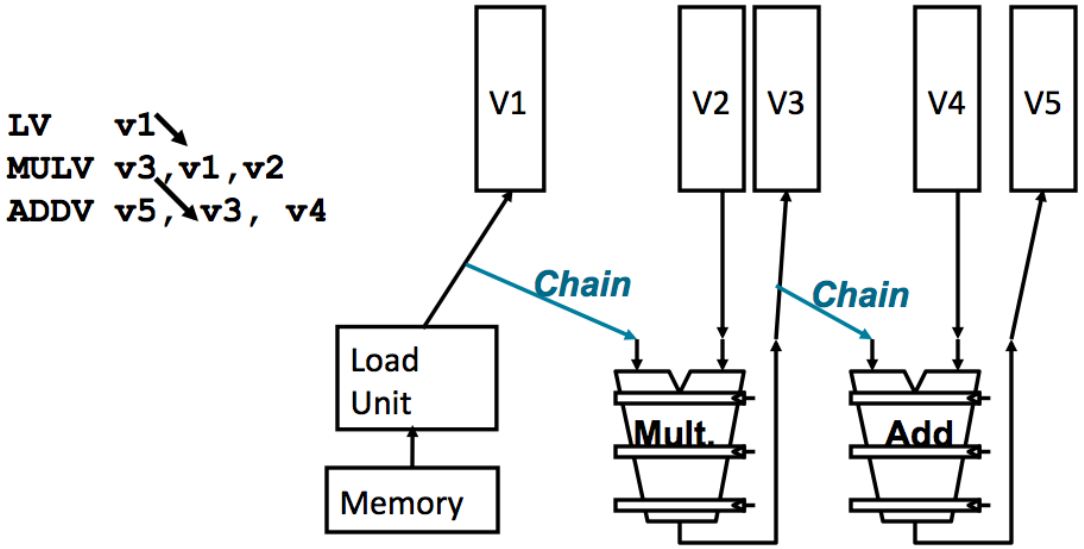
Vector Chaining

Input operands are used as soon as they arrive from memory

- Avoids stalling until the entire vector register is filled

Results can be sent from one functional unit directly to another

- A vector version of register bypassing and forwarding



Vector Chaining

Chaining overlaps the loading of operands into vector registers
- The next operands are read in parallel with the use of the previous set

The vector length register (VLR) specifies number of reads

Vector elements do not have to occupy adjacent memory cells
- Unlike with multi-media extensions (MMX, SSE and AVX)
- *Stride* = amount by which elements are separated in memory

Results can be chained back to memory as they are produced

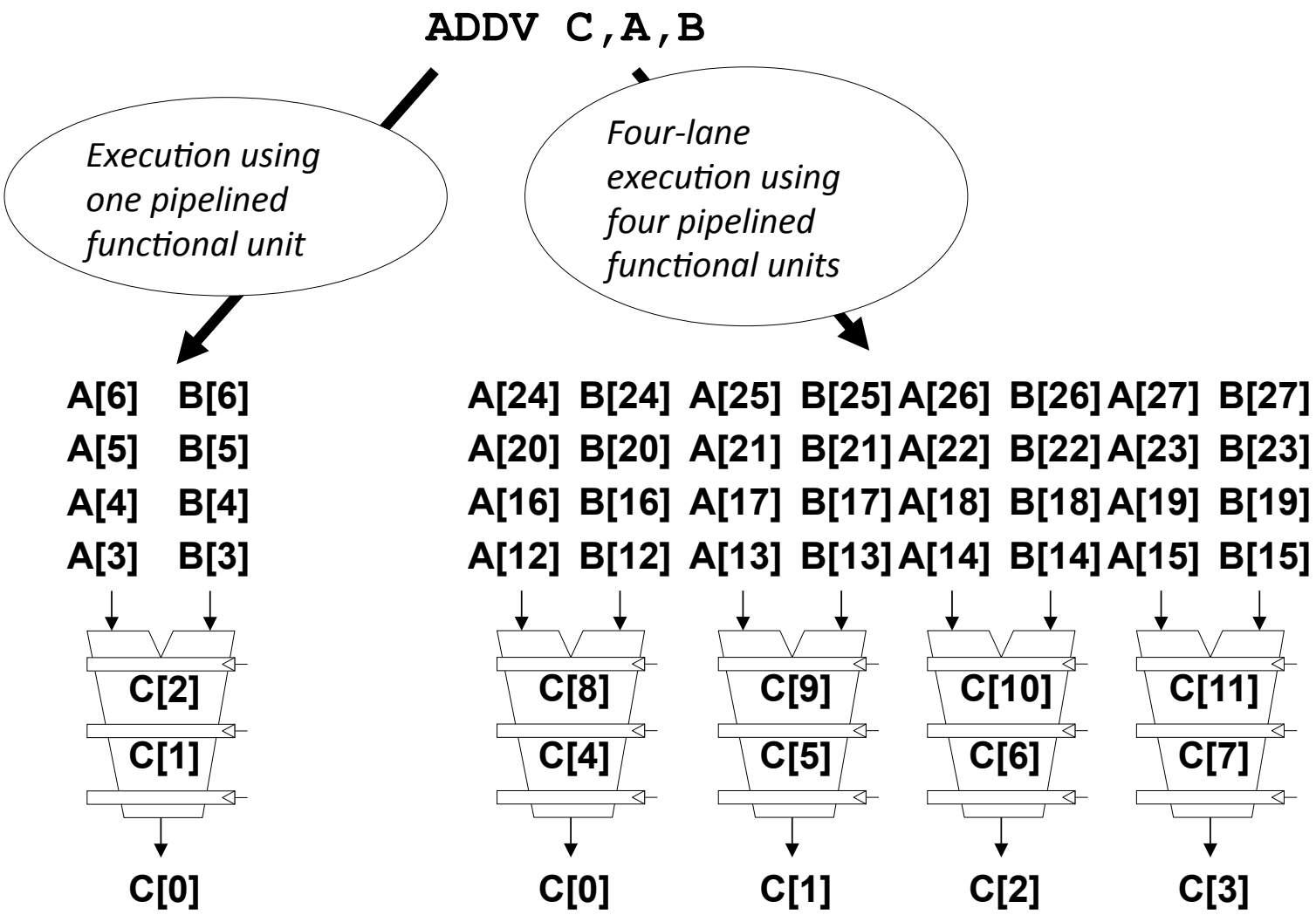
Scatter/ gather

Some vector processors use registers to locate data items
- vector registers need not be loaded from adjacent locations
- These indexed loads are called a *gather* operation

Indexing via registers can also be used to store vector results
- Elements in vector registers are dispersed to non-adjacent locations
- This is known as a *scatter* operation

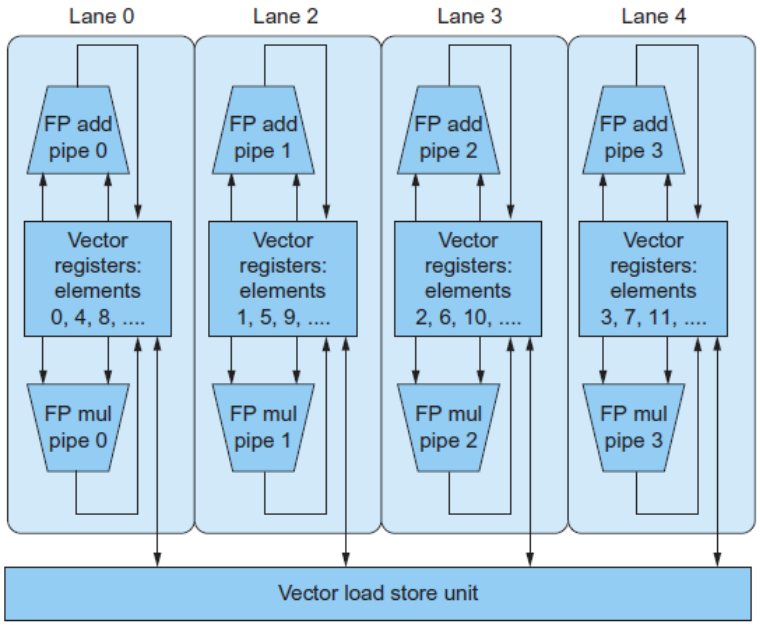
Strip mining compensates for mismatch in vector register length
- E.g. if vector is of length 60 and register is of length 32
- Vector register is loaded with first 32 elements
- And VLR (vector length register) is adjusted to load the next 28

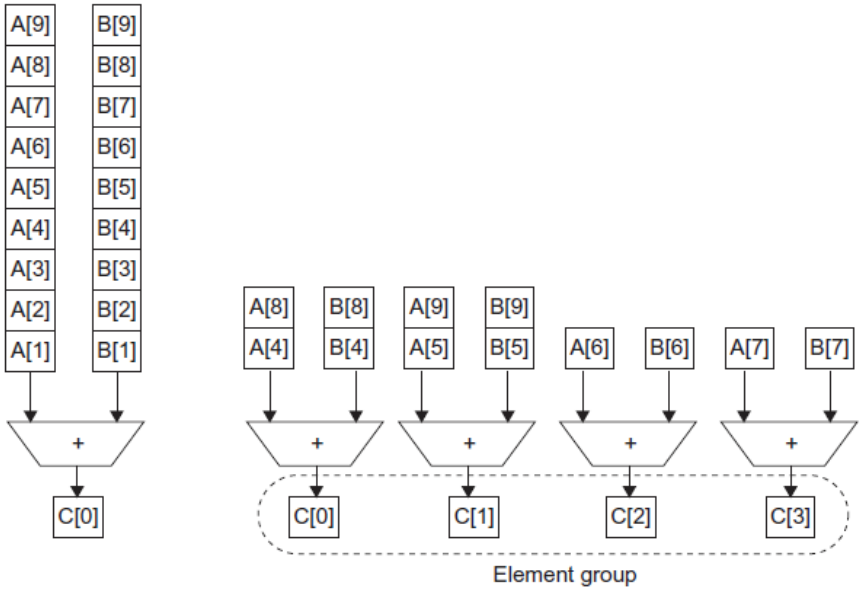
Example Vector Addition



Vector Processors

- Element n of vector register A is "hardwired" to element n of vector register B
  - Allows for multiple hardware lanes
  - No communication between lanes
  - Little increase in control overhead
  - No need to change machine code

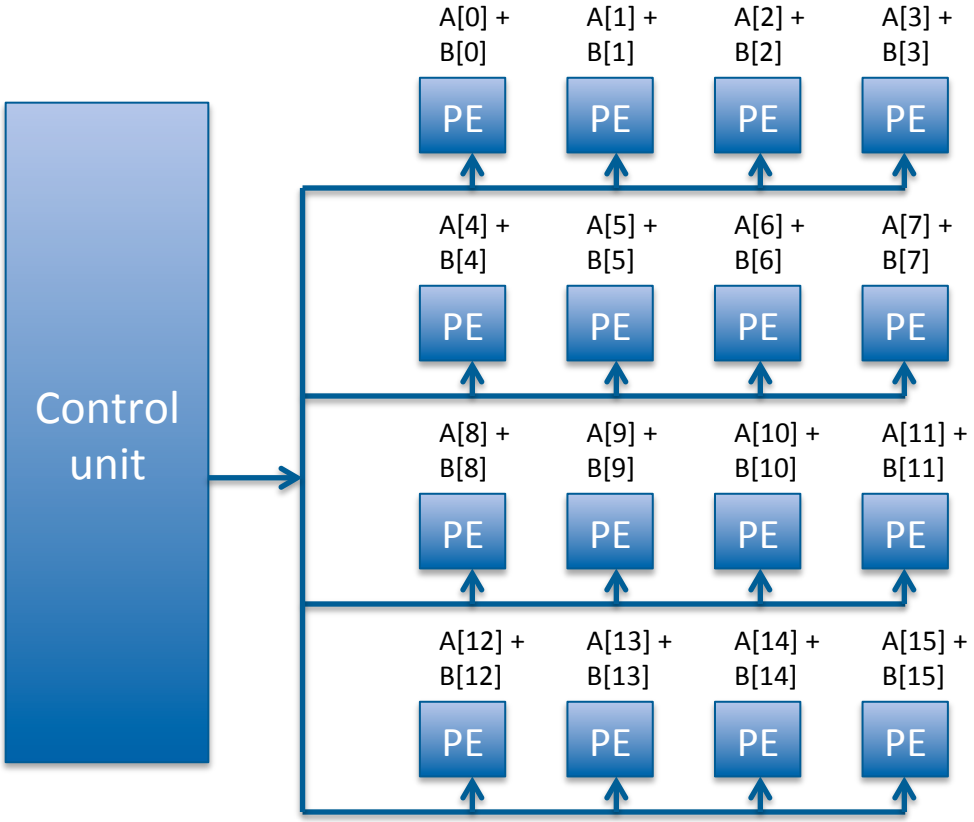




Adding more lanes allows designers to tradeoff clock rate and energy without sacrificing performance!

Array Processors

- An alternative in an array processor (SIMD)
- Separate processing elements could add corresponding array elements



One control unit broadcasts the same command to multiple processing elements that operate in lock-step fashion.