Vector Microprocessor

PhD Thesis, Krste Asanovic

Phani Jayanth J

Why Vector Architectures?

- Performance enhancement by exploiting “Instruction-Level Parallelism” was predominantly studied throughout the years. However, the improvement has not been significant - Large memory latencies being a major reason.
- Different workloads require specialized architectures to achieve gains in performance, speed, and energy.
- Vector Architectures offer an inexpensive way (since data parallelism is rigid) of achieving high performance on data parallel programs.
- Vector machines can saturate the currently available memory bandwidths with minimal control logic complexity and thereby can provide higher throughput.
- **Primary Constraint** - The Vector Functional Unit (VFU) Throughput decides the ultimate performance of the vector machine.
- Vector Architectures can be used in Multimedia, AI, and Scientific applications.

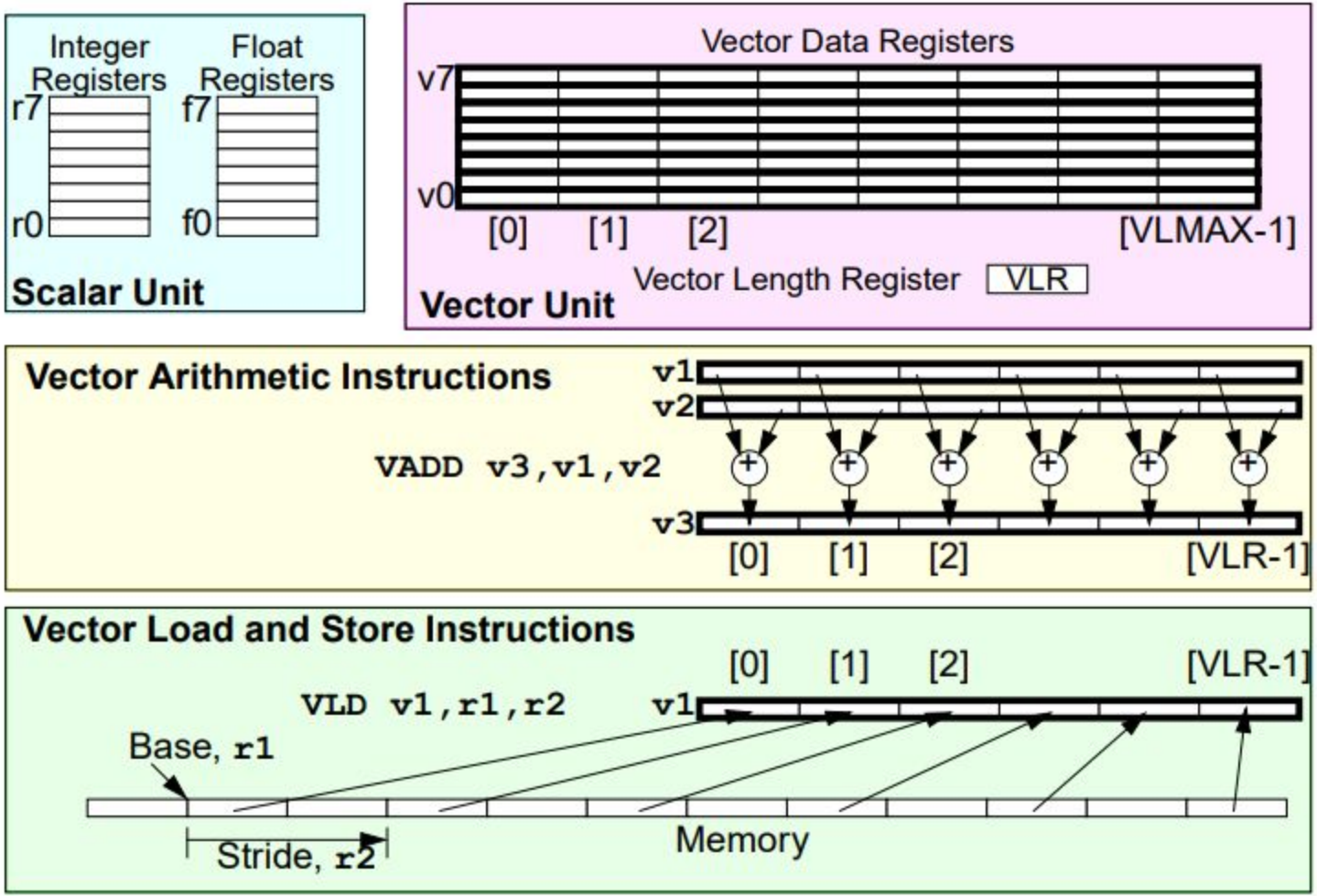
Vector Processing

- A Vector Instruction specifies vector operands, a vector length, and an operation to be applied element-wise to the vector operands.

$$\text{Vector ADD: } C_i \leftarrow A_i + B_i \ (i = 0,1,\ldots,VL-1)$$

$$VL = \text{Vector Length}$$

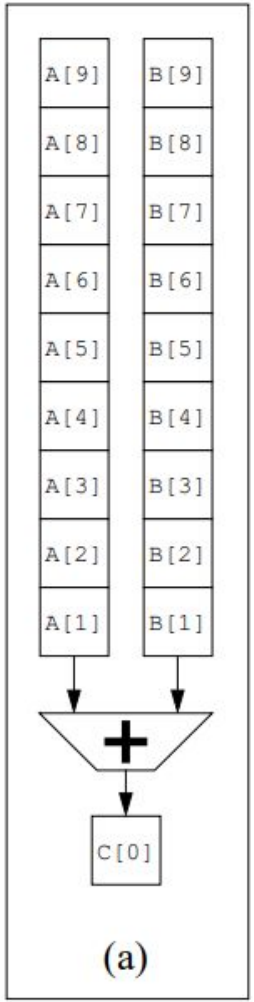
- Adding Vector Instructions also requires special Vector **Load-Store** Instructions that efficiently utilize the available memory bandwidth (Stride and Index-based vector accesses are crucial features; #Ports to the Vector Register File (RF) must be chosen carefully).
- **VLMAX** - Maximum Vector Length, **VLR** - Operand Vector Register Length (Set by the Programmer - Between 0 and VLMAX).
- Vector Flag Registers, Vector Mask Registers, etc., are also required for specific operations.

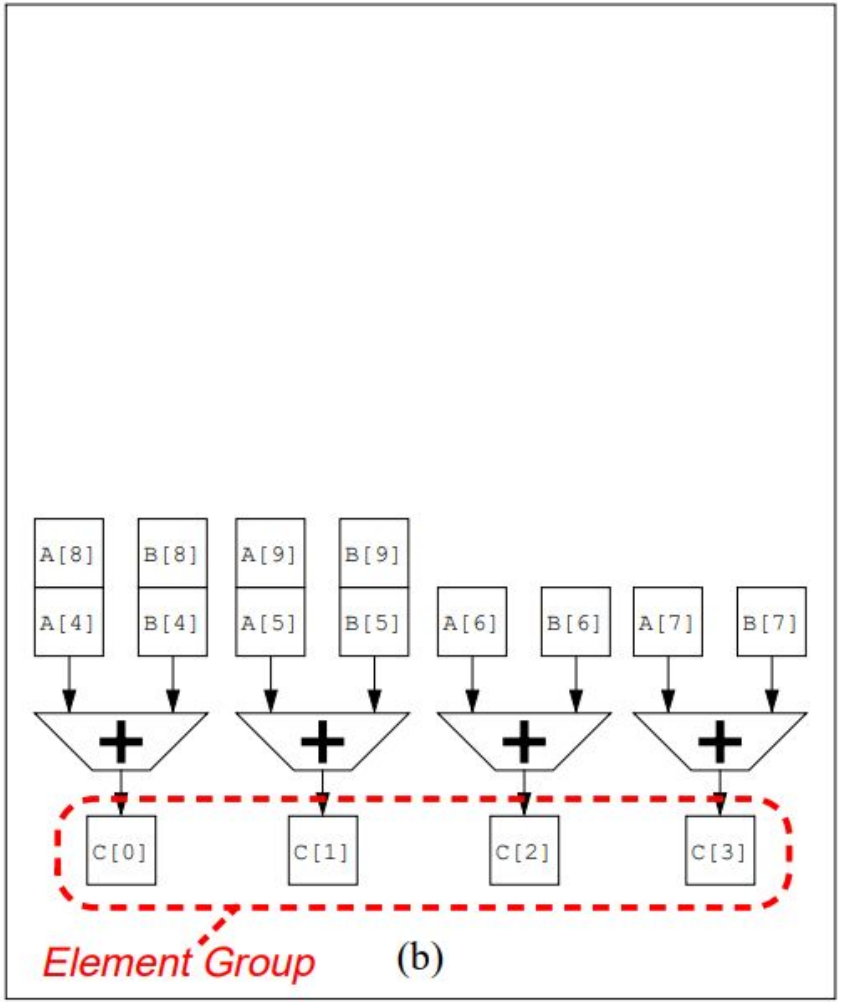

Integer Registers
Float Registers
r7
r0
f7
f0
Scalar Unit
Vector Data Registers
v7
v0
[0]
[1]
[2]
[VLMAX-1]
Vector Length Register
VLR
Vector Unit
Vector Arithmetic Instructions
VADD v3,v1,v2
v1
v2
v3
[0]
[1]
[2]
[VLR-1]
Vector Load and Store Instructions
VLD v1,r1,r2
Base, r1
Stride, r2
Memory

Advantages and Key Points

- These Vector Instructions comprise multiple **homogenous**, **independent** operations - compacted to form **expressive** and **scalable** programs.
- For a Vector Instruction describing **N** operations: **3N** register operands
  - **N Homogenous** operations.
  - **N Independent** operations.
  - Will access elements in the vector operand registers in a regular pattern.
  - Instruction itself operates in a known pattern.
- Performance can be increased by adding parallel pipelines (at the cost of area).
- Throughput increases linearly with the number of pipelines in the VFU.
- Vector Instructions can be added as extensions to existing scalar ISA.
- **Vector Architecture and Software**
  - High-performance, low-cost, and energy-efficient for data parallel tasks.
  - Portability across a range of different vector machines.
  - Scalability with increased number of parallel pipelines

Single Pipeline VFU





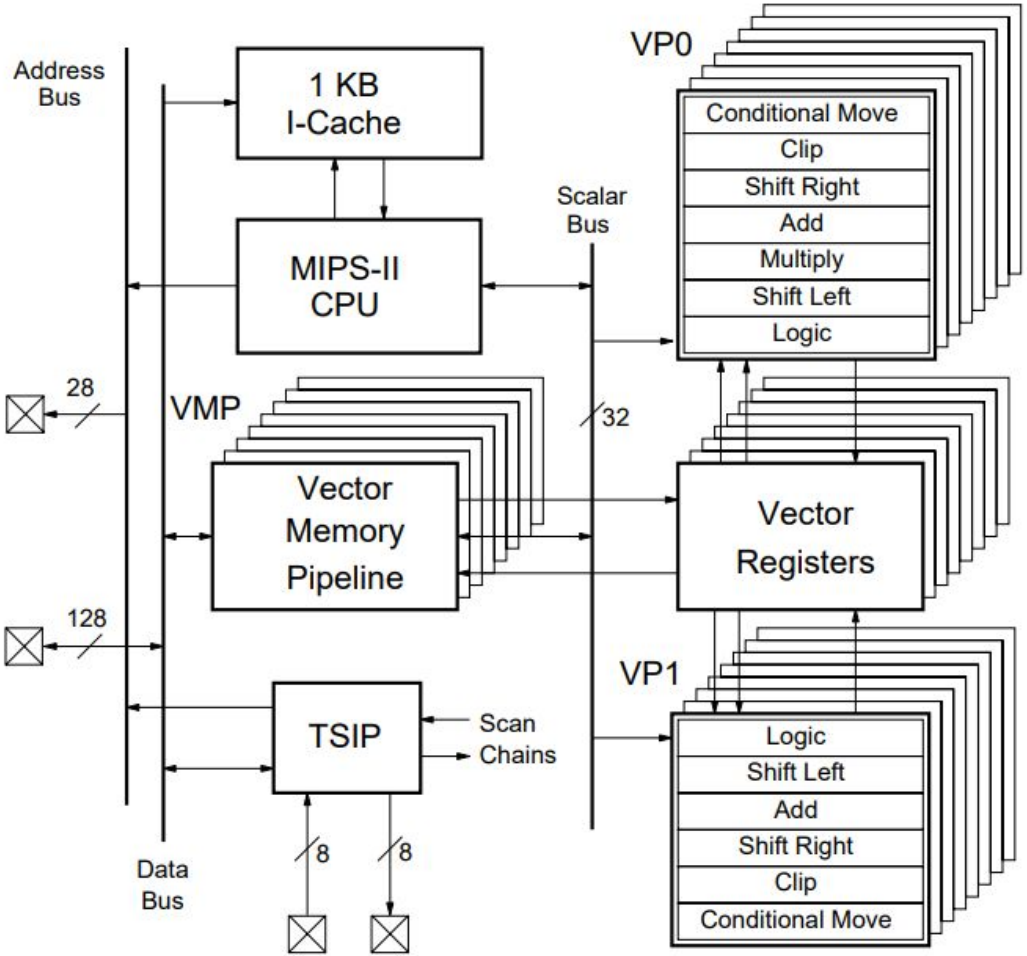
Four Pipeline VFU

Torrent-0 Microprocessor

- The T0 Vector Microprocessor is developed within ICSI, and later in collaboration with U. C. Berkeley, to train neural networks for phoneme probability estimation in speech recognition research.
- Planned to implement a vector coprocessor attached to MIPS R3000 Core **(Discarded)**
- **Why the Coprocessor Idea was discarded?**
  - Complicated overall system design (particularly of the memory system)
  - Vector Coprocessor would require its own wide, fast, and coherent connection to the memory alongside the cache system used by the scalar unit - considerably adds to size, complexity, and cost.
  - Coprocessor package also requires more pins to connect to both the processor cache bus (for instructions) and the memory system (for data).
  - Loose coupling between vector coprocessor unit and the scalar CPU.
  - Speed of inter-chip signal paths.
- T0 builds upon the scalar MIPS ISA - with 16 Vector Registers and a VLMAX of 32 with each element holding 32 bits.

T0 Microarchitecture

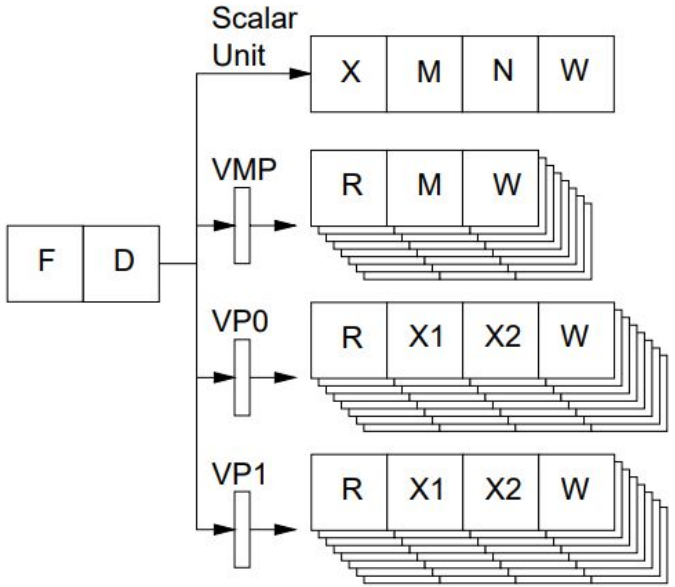
- T0 integrates a scalar unit, 1KB Instruction Cache, the Vector Unit, external memory interface, and a serial host interface port (TSIP).
- The memory interface has 128 bit data bus - supporting up to 4GB async. SRAM.
- The TSIP has 8-bit wide datapaths, and provides DMA into T0 memory as well as access to on-chip scan chains for testing and debugging.
- **The Vector Unit** consists of **three VFUs** - **VMP, VP0**, and **VP1** - communicate via the Vector Register File (VRF - with 16 Vector Registers).
- **VMP** - Vector Memory Unit - Executes Scalar Memory and Vector Extract Instructions and Vector Memory Instructions.
- **VP0 and VP1** - Vector Arithmetic Units (Only VP0 contains a multiplier).
- **All 3 VFUs** contain **eight parallel pipelines** and therefore can produce up to eight results per cycle.
- Furthermore, the vector unit is structured into **eight parallel lanes** - each lane contains a portion of the VRF and one pipeline for each VFU
- **Note: A well-defined vector ISA constraints most communication between vector instructions to occur locally within a lane.**



T0 Architecture
Block Diagram

Fetch and Decode

- Fetch and Decode Stages are common to all the instructions.
- Fetch (F) Stage accesses the 1KB direct-mapped instruction cache. The next instruction address is also determined in the F stage and fed to the cache.
- F Stage operates independently of later stages - cache miss processing can overlap with interlocks and memory pipe stalls.
- During the Decode (D) Stage, the scalar register is read and bypassed and interlocks are checked. If there are no interlocks, the instruction is dispatched to the appropriate execution unit. Otherwise, it is held in the D stage until interlocks clear,
- Scalar pipeline handles exception checking.
- Vector Instructions can occupy the VFUs for multiple cycles.



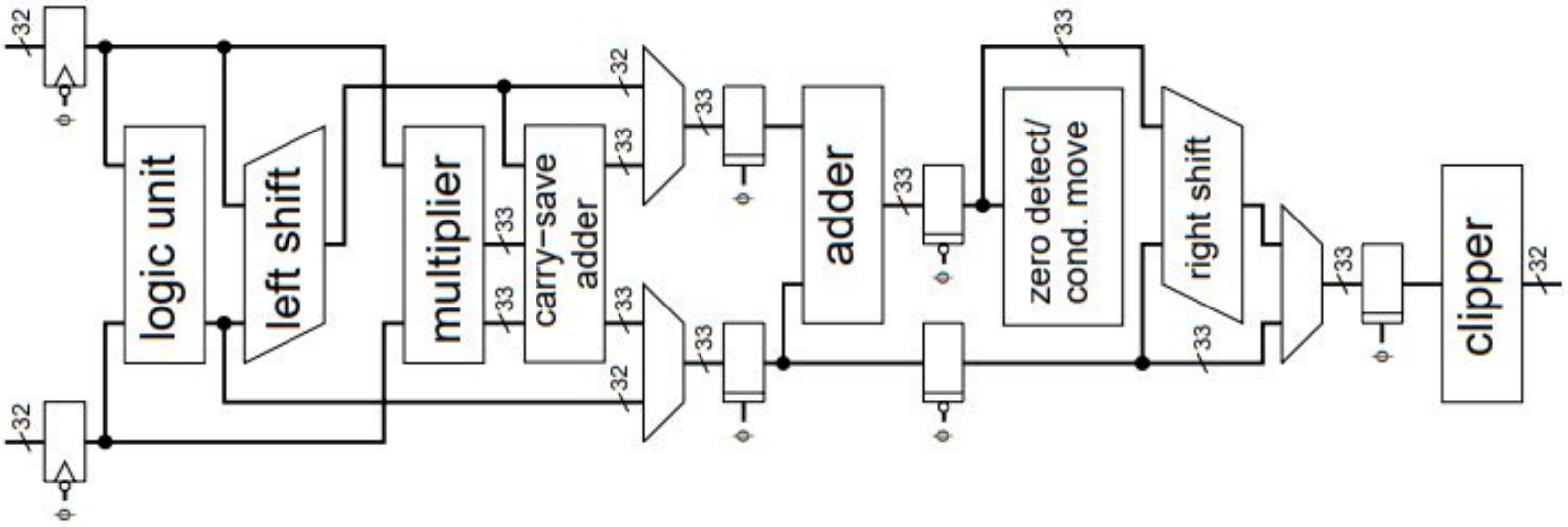
VRF and VMP

- The T0 VRF contains 16 Vector Registers, each holding 32 32-bit elements.
- Vector Register 0 is hardwired to the value zero.
- The VRF is split into eight parallel 32-bit wide slices, one for each of the lanes.
- Each lane contains a total of 60 READ/WRITE 32-bit registers.
- The VRF contains one read and one write port for VMP, two reads and one write port for each of VP0 and VP1 - total of five read ports and three write ports.
- To save area, the register file bit cells are time-multiplexed (with increased complexity of control logic).
- The address decoders for the VRF are also time-multiplexed to reduce area.
- The Vector Memory Unit contains three stages - VRF Read (R), Memory Access (M), and VRF Write (W).
- In R stage, VRF Read Address is calculated and passed to the decoders. And then, the VRF is read.
- Scalar Stores are always done on Lane 0.
- In M Stage, Store Data is aligned and passed to the external memory. For a Load, data is returned from external memory, and aligned properly and latched into the registers.
- During the W stage, the latched load data is aligned by suitable extension and written to the VRF.

Vector Arithmetic Units (VAU)

- VP0 and VP1 together allow T0 to sustain sixteen 32-bit ALU operations per cycle.
- Arithmetic Instructions are dispatched to VAUs at the end of the D Stage.
- VAU pipelines operate on 32-bit data, taking two 32-bit operands and producing a 32-bit result plus a single bit flag value.
- Each VAU supports 32-bit integer arithmetic and logical operations, with fixed-point arithmetic support.
  - Each pipeline contains 6 functional units: 32-bit Logic Unit, 32-bit Left Shifter, 33-bit Adder, 33-bit Shift Right Unit, 33-bit Zero Comparator, and a 33-bit Clipper.
  - VP0 contains a 16-bit X 16-bit multiplier additionally.
- Data flows through all these functional units for all instructions - unused functional units will be set to pass the data unchanged - This allows multiple operations to be performed with a single pass through the pipeline (adds way for special instructions).
- For operations with narrower elements, the vector unit can be considered to have more narrower virtual processors (VPs) - a vector machine with 32 64-bit VPs can also be treated as 64 32-bit VPs or 128 16-bit VPs or 256 8-bit VPs.
- **Maximum Vector Length now becomes a function of VP width.**

A Single Pipeline of VAU



Variable Width Virtual Processors

- Narrower subwords can be elegantly handled with vector ISA.
- For operations with narrower elements, the vector unit can be considered to have more narrower virtual processors (VPs).
- For example, a vector machine with 32 64-bit VPs can also be treated as 64 32-bit VPs or 128 16-bit VPs or 256 8-bit VPs.
- **Maximum Vector Length now becomes a function of VP width.**
- VP Width can be specified in every instruction or by writing to a separate status word.
- A VP can load and store all data sizes less than or equal to its width.

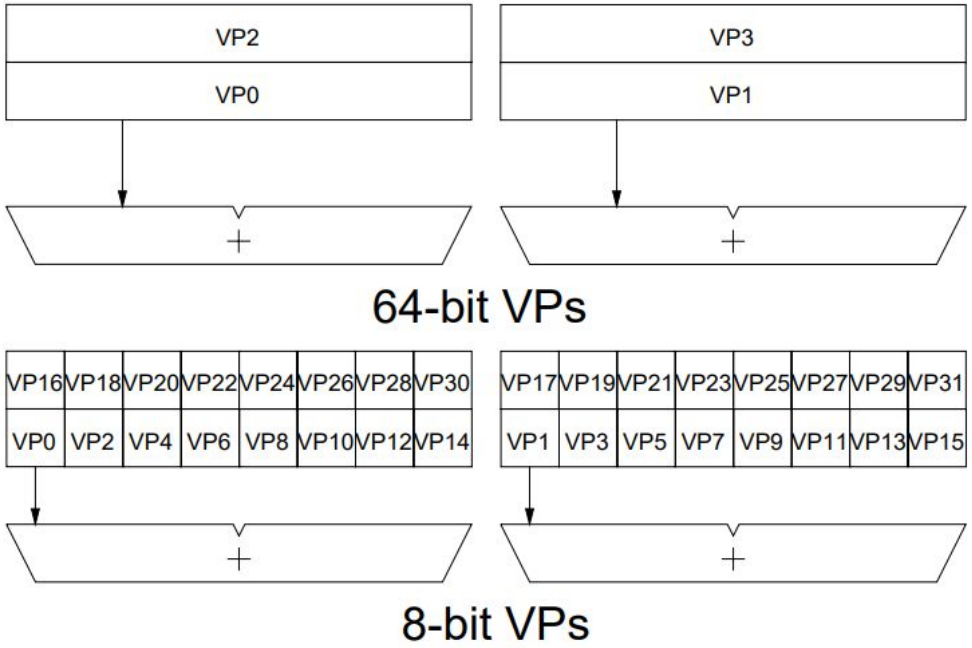


Figure 7.2: Mapping of VPs to two 64-bit datapaths when 64-bit address generators are located one per 64-bit lane. The upper figure shows how 64-bit VPs are mapped to the datapaths, while the lower figure shows 8-bit VPs. Both 64-bit and 8-bit VPs will run strided and indexed operations at the rate of 2 elements per cycle.

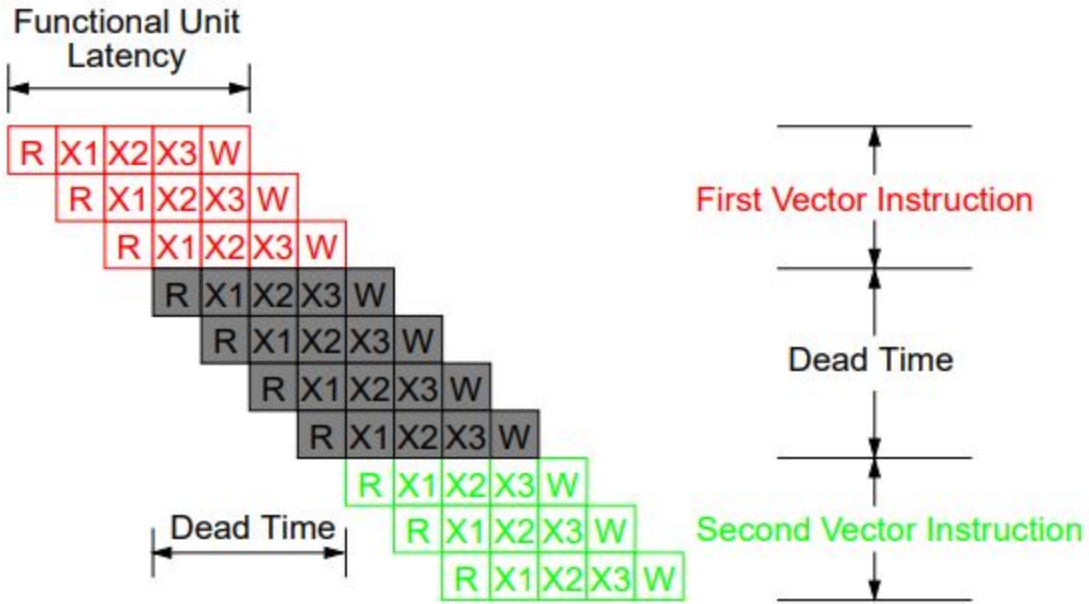


Figure 3.10: Pipeline diagram of one vector functional unit showing the two components of vector startup latency. Functional unit latency is the time taken for the first operation to propagate down the pipeline, while dead time is the time taken to drain the pipelines before starting another vector instruction in the same vector functional unit.

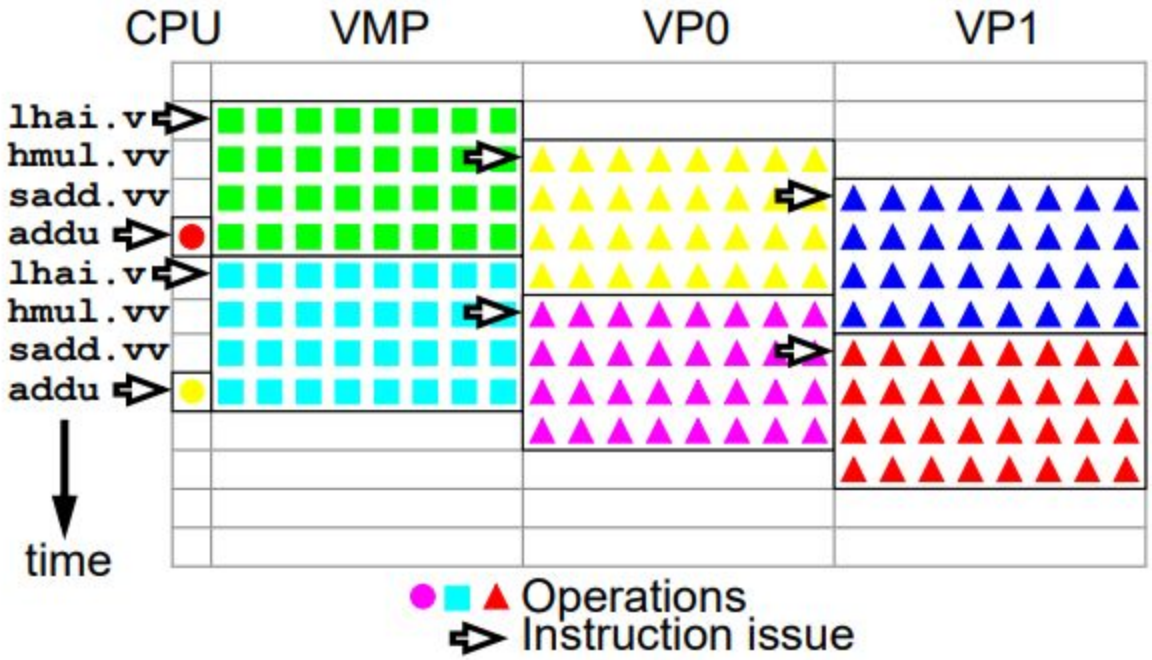


Figure 3.11: Execution of a code fragment on T0 with vector length set to 32 elements. A vector load is issued to VMP on the first cycle, and begins execution of eight operations per cycle for four cycles. A vector multiply is issued on the subsequent cycle to VP0, and the execution of this vector multiply overlaps with the vector load. Similarly, a vector add is issued on the third cycle to VP1, and execution of the add overlaps the load and multiply. On the fourth cycle a scalar instruction is issued. On the fifth cycle the vector memory unit is ready to accept a new vector load instruction and the pattern repeats. In this manner, T0 sustains over 24 operations per cycle while issuing only a single 32-bit instruction per cycle.

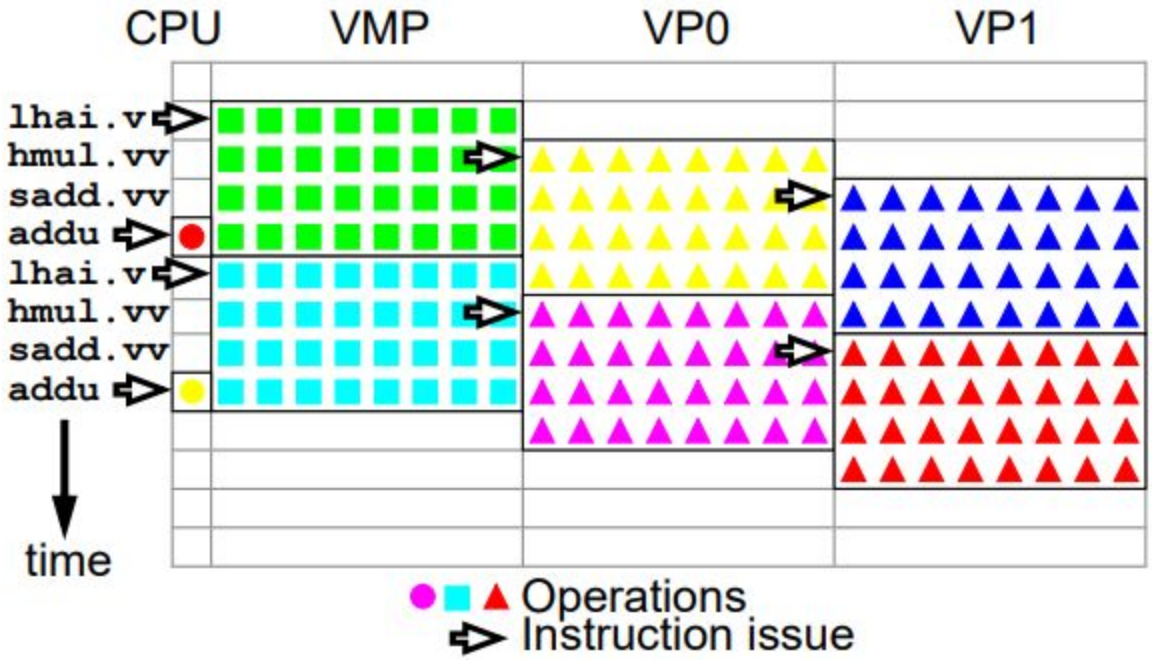


Figure 3.11: Execution of a code fragment on T0 with vector length set to 32 elements. A vector load is issued to VMP on the first cycle, and begins execution of eight operations per cycle for four cycles. A vector multiply is issued on the subsequent cycle to VP0, and the execution of this vector multiply overlaps with the vector load. Similarly, a vector add is issued on the third cycle to VP1, and execution of the add overlaps the load and multiply. On the fourth cycle a scalar instruction is issued. On the fifth cycle the vector memory unit is ready to accept a new vector load instruction and the pattern repeats. In this manner, T0 sustains over 24 operations per cycle while issuing only a single 32-bit instruction per cycle.