SR8000 – Pseudo−Vectorization

Pseudo−vectorization	Pipelining	Sliding registers, predicates and other things.
	Hiding latency	**The three ways to memory:** Prefetch a cacheline Preload 8 bytes to register Load a cacheline

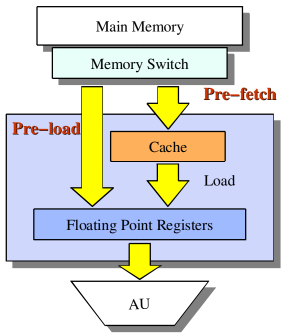
Since this seems to be a common misunderstanding:
Pseudo−vectorization here has **nothing** to do with parallelization

Pipelining	**Vectorizable code**. **Advice**: Make use of predicates, inlining, speculative prefetches and vectorized math routines. If your code is direct access the combination of compas and prefetches should be your method of choice. Bandwidth restrictions and the FPU (2 muladds) will limit your peak performance to 4 Gflop/node if you can't re−use your data. **Advice**: Make use of loop unrolling if you can cache−block the data. The SR8000 is both – vector and cache architecture.

Pipelining	**Vectorizable code**. **Advice**: If your code is vectorizable, but with indirect access the method of choice strongly depends on wether the code can be cache blocked. If you need to access the memory in completely arbitrary fashion with no re−use of data you need to rely on preloads. **Note**: The bandwidth for preloads is only half of the regular bandwidth through the cache. In the above (random access) case you won't see more than 2 Gflop/node.

Hiding latency	**Other codes.** The combination of preloads/prefetches and cache hits can be a very powerful one. Try to keep the vectorizable parts seperated from non-vectorizable parts. Use compas section based parallelization, OpenMP or MPI. **Advice:** If you can keep your data localized with indirect access it might be worth to switch of preloads via compiler directives.

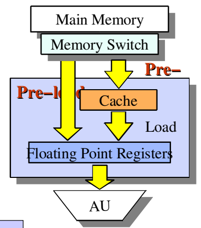
Architecture



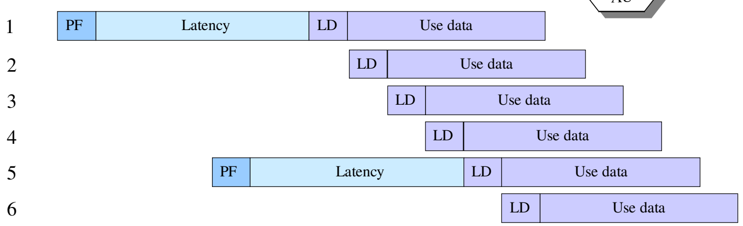
- 128 Kbyte L1 cache
- 16 bytes/cycle memory BW
- Pre-fetch and pre-load instructions
- 160 f.p. registers
- 2 f.p. Pipelines
- 4 flops/cycle

- Pre-fetch: load cache line from memory to cache
- Pre-load: load one word from memory to register
- 16 streams a(i,j), b(i,j), b(i,j+1)

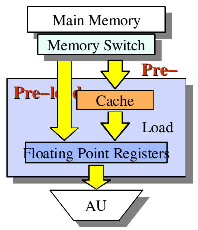
Prefetch



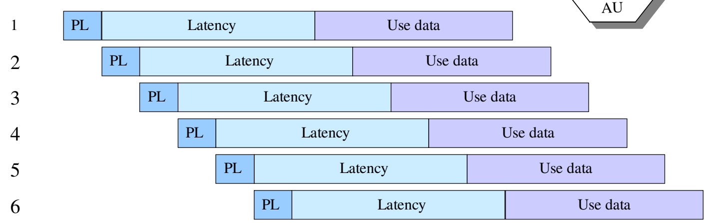
- Pre-fetch 128 bytes to cache
- Follow by LD to register



Preload

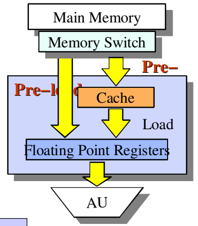


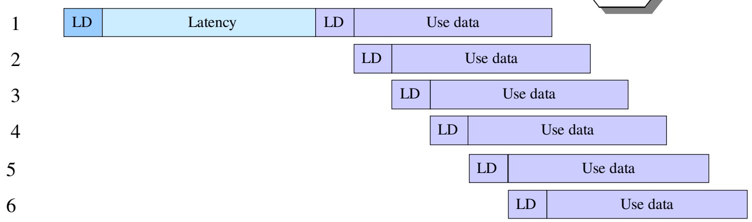
- Pre-load 8 bytes to register
- LD not required



Load

- Load 128 bytes to cache
- Load to register
- Exploit cache hits





Pseudo-vectorization

$$A(:) = A(:) + N$$