INF5063:
Programming heterogeneous multi-core processors

SIMD

September 24, 2010

SIMD

- SIMD: single instruction multiple data
  - low instruction bandwidth
  - cheap synchronization
  - easy addressing of memory blocks
  - pipelining or prefetching operations

- classical approach
  - long vectors
  - a recent processor: NEC SX-9
  - 16 mask registers à 64 bit
  - 72 vector registers á 4096 bits (512 bytes)
  - "assignable data buffer" in-processor memory

- recent approach
  - short vectors
  - a recent architecture: Intel 64
  - 8 XMM (general purpose SIMD) registers à 128 bits
  - prefetch to cache, asynchronous write, write instruction barrier

SIMD

- success of short vectors
- fits well to image processing
  - ray tracing
  - color space conversion
  - DCT and FFT computation
  - fractal sets (Mandelbrot ...)
- and is not bad for some other ops
  - matrix multiplication
  - parallel state machines
  - numerical PDE solvers

SIMD: typical use

- "SIMDification"
  - unlike true parallel architectures
  - look for repetitive use of the same instruction
  - look locally ("peephole optimization")

- convert a "normal" program to use SIMD instructions
  - identify and extract parallelism in code
  - satisfy constraints

SIMD: typical use

- search for
  - constant-length short loops
    - unroll completely
    - implement with SIMD instructions
  - loop-level parallelism
    - unroll partially
    - implement with SIMD instructions
  - basic-block parallelism
    - don't touch

SIMD: limitations

- reading from and writing to memory
  - aligned read and write
  - contiguous read and write
  - no strided read and write
  - replaced by helpers for "streamshifting" data

- conditions and conditionals
  - no flag registers
  - carry, overflow, ... independently for each block
  - conditional execution replaces conditional branches

SIMD: compilers' challenges

- unroll and reorder across several operations
- handling of unaligned arrays and fields in structs

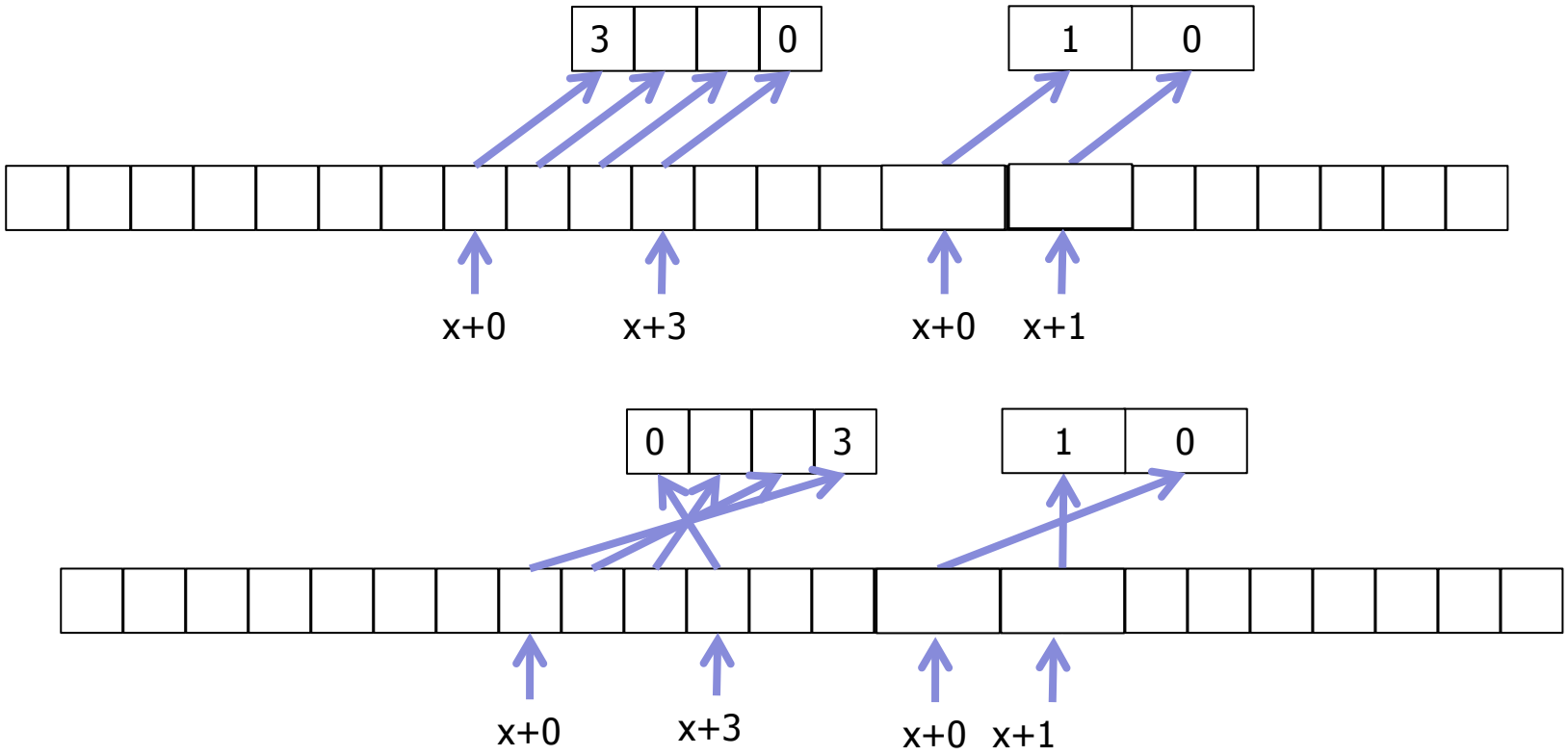
```
struct dataset
{
 vector unsigned short a;
 unsigned int onebit:1;
 vector unsigned int b;
};

code()
{
 struct dataset x;
 vector unsigned int* mine = (vector unsigned int*)((char*)&x.onebit+1);
 ...
}
```



```
QUAD q;
```

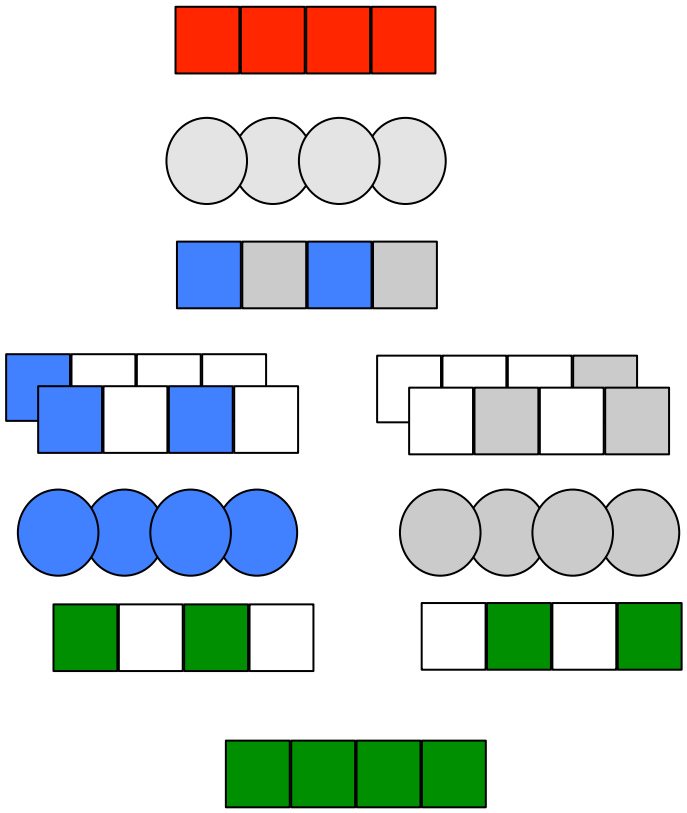
SIMD: compilers' challenges

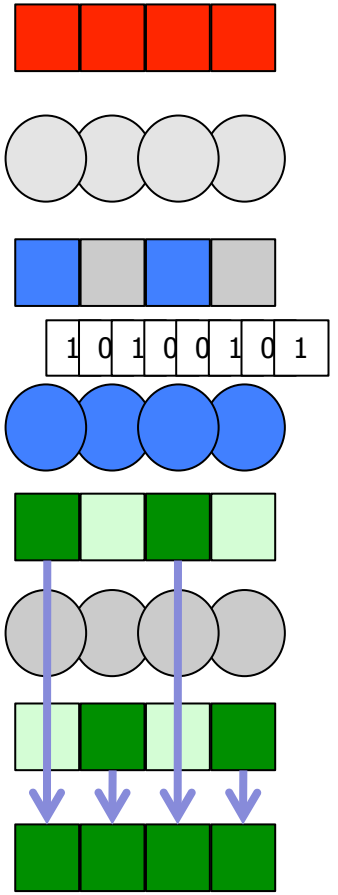
- abstract alignment
  - for big endian: indexing in memory vs. in vector register
  - for little endian: data ordering in memory vs. in vector register



SIMD: compilers' challenges

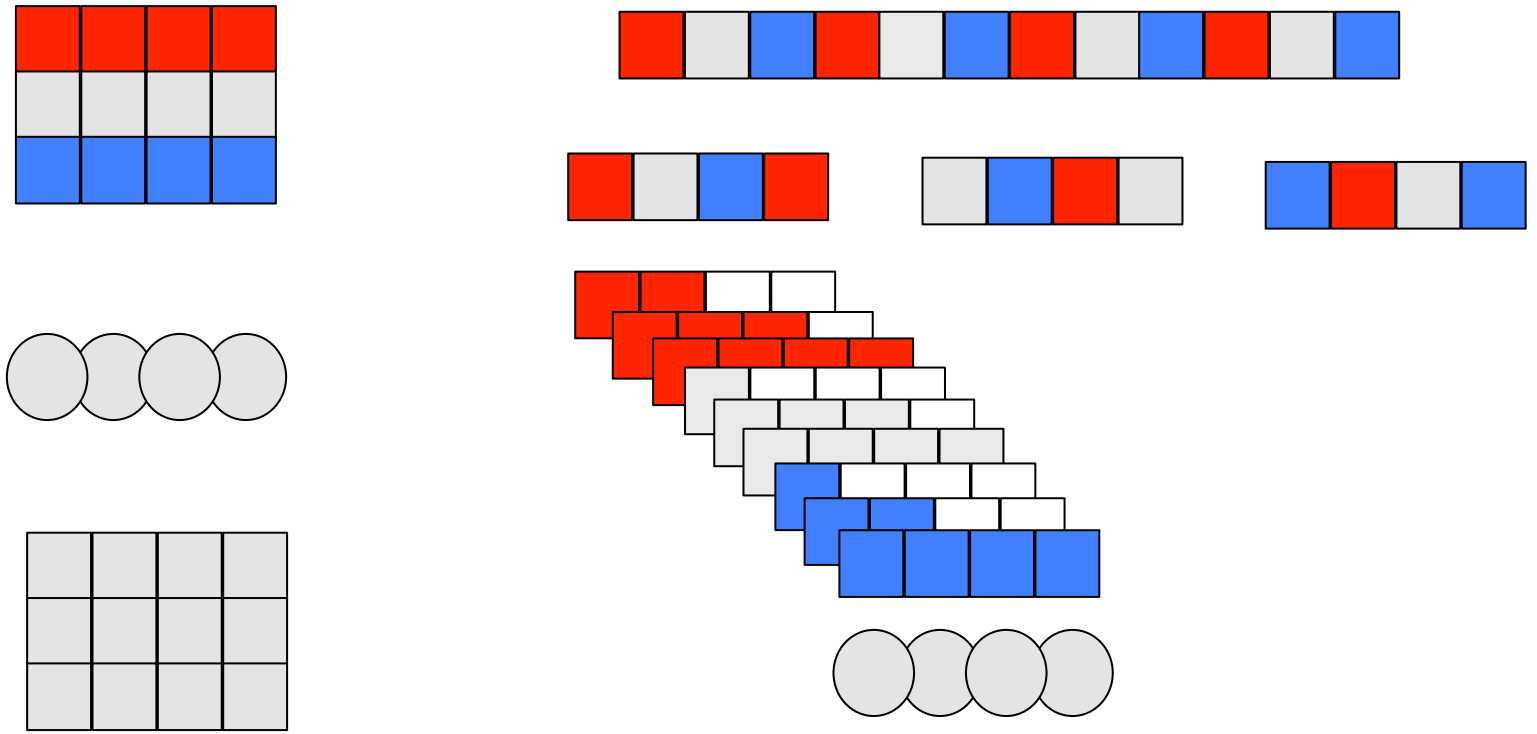
- partial storage of results
  - write with mask
- conditional execution vs. branching





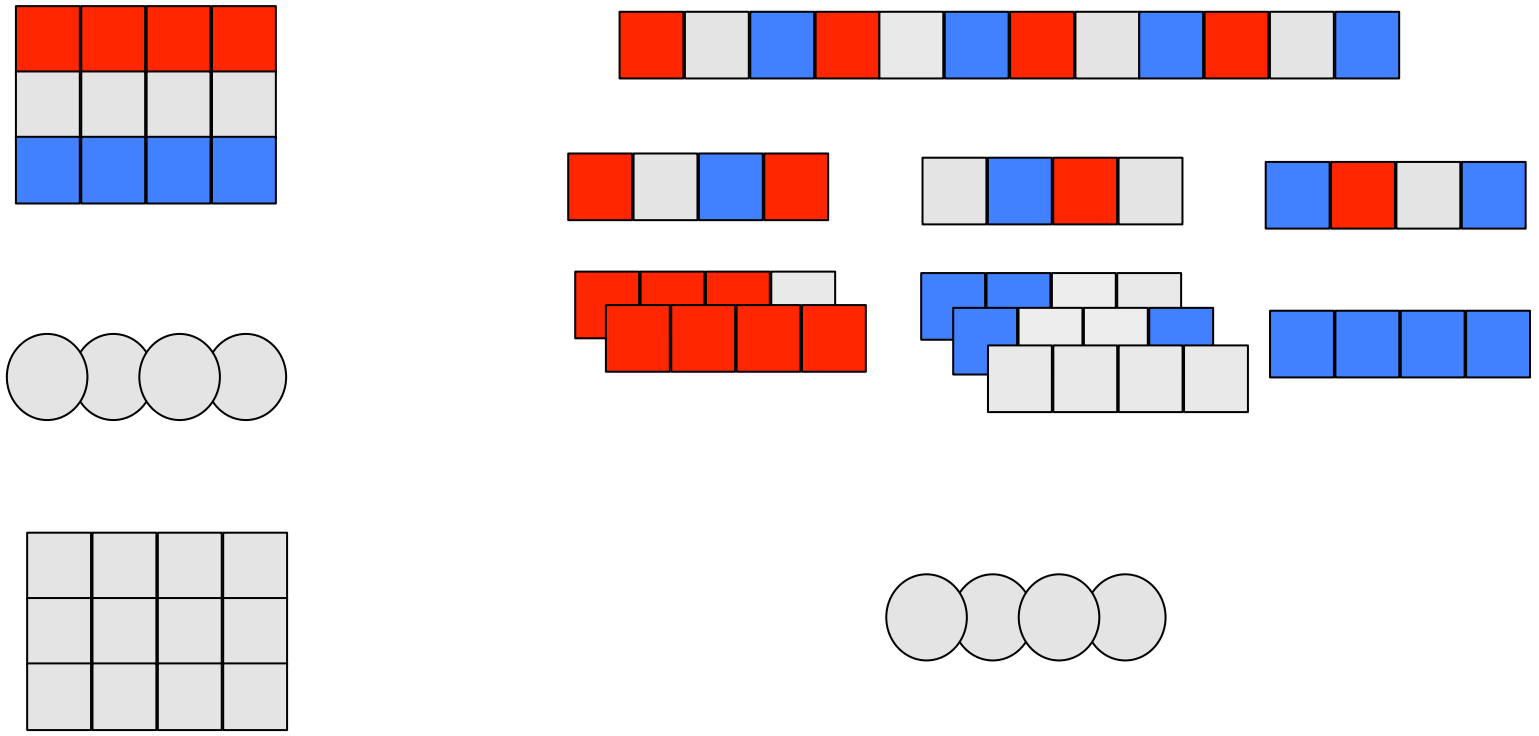
SIMD: compilers' challenges

- data tracing
  - commuting and merging instructions



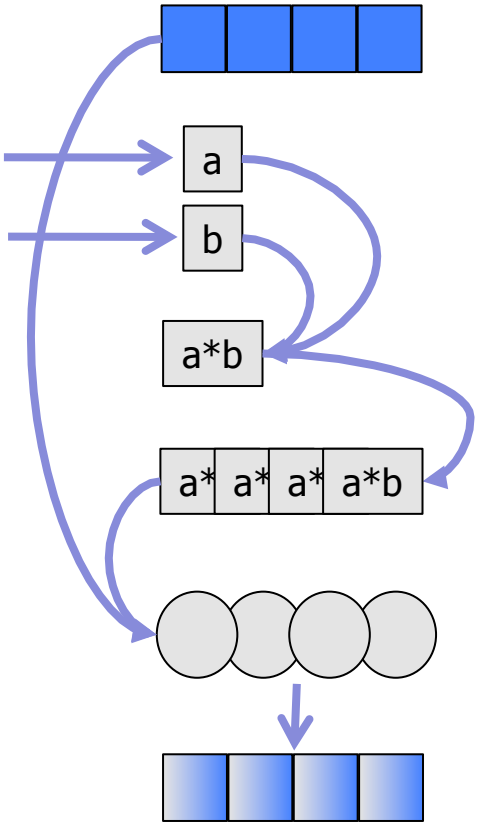
SIMD: compilers' challenges

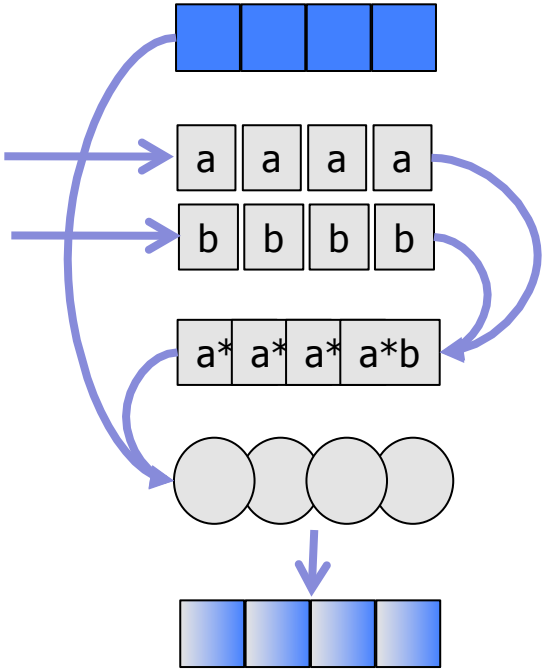
- data tracing
  - commuting and merging instructions



SIMD: compilers' challenges

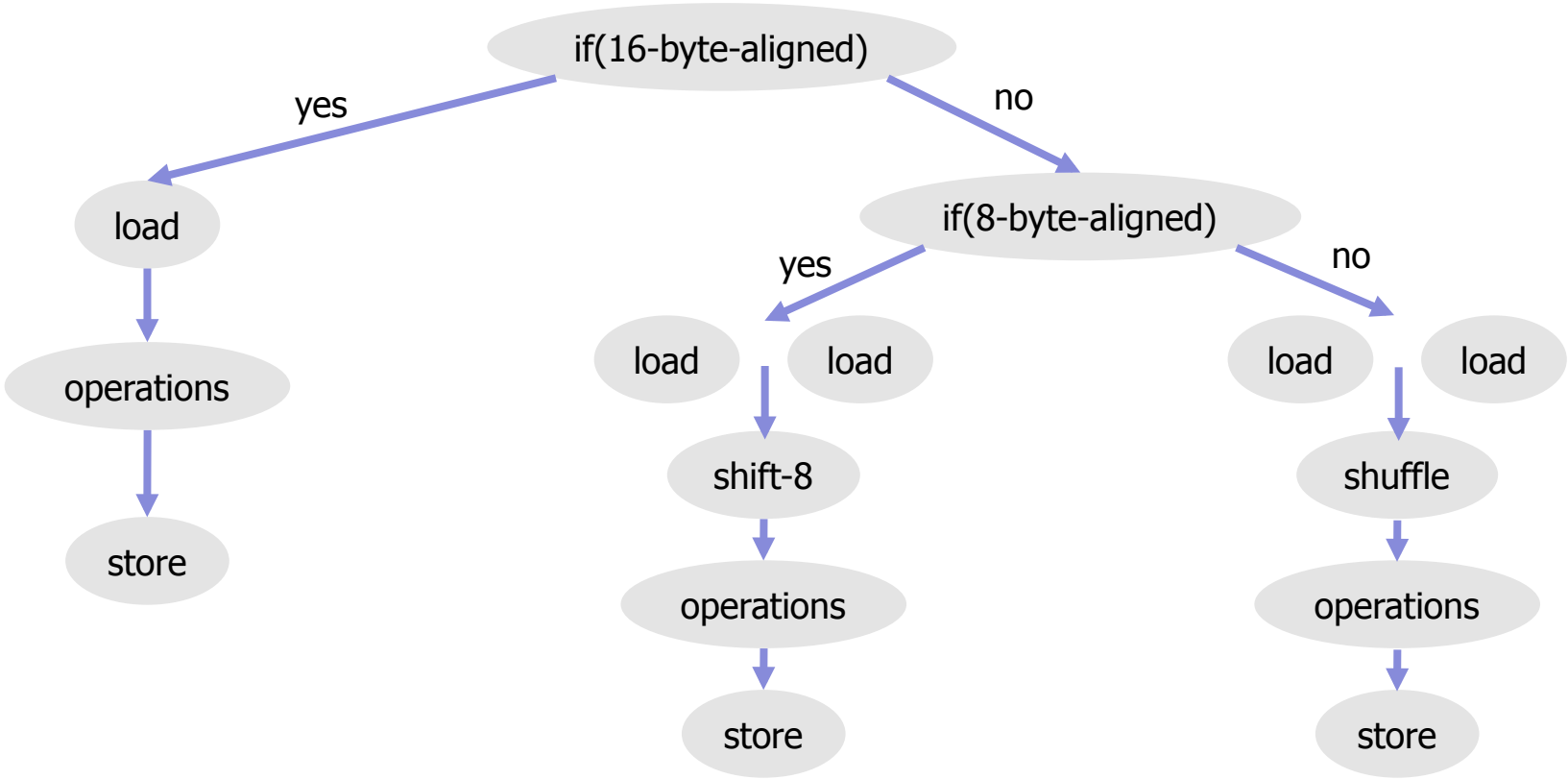
- data tracing
  - replicating instructions





SIMD: compilers' challenges

- handling alignment in the dynamic case
  - compiler must build a "shuffle tree" (data reorganization graph)
  - make run-time decision about the shuffle branch



SIMD compiler builtin function: AltiVec

- Arithmetics: vec_abs vec_add* vec_sub* vec_m*add* vec_msum* vec_nmsub vec_sum*s vec_and* vec_or vec_nor vec_xor vec_avg* vec_ceil vec_floor vec_round vec_trunc vec_loge vec_max vec_min vec_mul* vec_rsqrte vec_expte vec_re

- Compare: vec_m*vscr vec_cmpb vec_cmpeq vec_cmpge vec_cmpgt vec_cmple vec_cmplt vec_all_<test> vec_any_<test>

- Shift, rotate, permutate (whole or parts): vec_sr* vec_sl* vec_rl vec_perm vec_splat vec_pack* vec_unpack* vec_merge* vec_lvsr vec_lvsl

- Selection: vec_sel
- Memory: vec_ld* vec_st* vec_dst vec_dstt vec_dstst vec_dststt vec_dss vec_dssall

SIMD compiler builtin function: Intel 64

- MXX & SSE (not AMD versions)
- all names are of the form __builtin_ia32_<name>

- Arithmetic: padd* psub* pmul* pand* por pxor pavg* pmax* pmulhuw psadbw pmaxsw pmin* add* sub* mul* div* max* min* and* orps xorps rsqrtss sqrtss rsqrtps sqrtps rcp*

- Comparison: pcmpeq* pcmpgt* cmp* comi* ucomi*

- Packing and shuffling: punpck* pack* unpck* shufps movmskps pmovmskb pextrw pinsrw

- Memory: mov* load* store* movntps maskmovq movntq sfence