# Prefetching

**Mike Bailey**

**mjb@cs.oregonstate.edu**

**Oregon State University**

# Prefetching



Prefetching is used to place a cache line in memory before it is to be used, thus hiding the latency of fetching from off-chip memory.

There are two key issues here:

1. Issuing the prefetch at the right time
2. Issuing the prefetch at the right distance

**The right time:**

If the prefetch is issued too late, then the memory values won't be back when the program wants to use them, and the processor has to wait anyway.

If the prefetch is issued too early, then there is a chance that the prefetched values could be evicted from cache by another need before they can be used.

**The right distance:**

The "prefetch distance" is how far ahead the prefetch memory is than the memory we are using right now.

Too far, and the values sit in cache for too long, and possibly get evicted.

Too near, and the program is ready for the values before they have arrived.

# Prefetching in g++



```
void __builtin_prefetch(  void *,  int rw,  int locality  );
```

```
#define WILL_READ_ONLY                  0
#define WILL_READ_AND_WRITE             1

#define LOCALITY_NONE                   0
#define LOCALITY_LOW                    1
#define LOCALITY_MED                    2
#define LOCALITY_HIGH                   3

#define PD                              32          // prefetch distance (fp words)

for( int i = 0; i < ArraySize; i++ )
{
        if(  (i%16)  == 0  )
        {
                __builtin_prefetch ( &a[ i+PD ], WILL_READ_AND_WRITE, LOCALITY_LOW );
                __builtin_prefetch ( &b[ i+PD ], WILL_READ_ONLY,      LOCALITY_LOW );
        }
        a[ i ] = a[ i ] + b[ i ];
}
```

# Prefetching in icc and icpc



Overall, icc and icpc seem to do a good job of prefetching without you doing anything extra, but if you want to be sure:

**#pragma prefetch var:which-prefetch:#vector-iterations**

```
#pragma prefetch a:0:8
#pragma prefetch a:1:32
#pragma prefetch b:1:32
for( int i = 0; i < ArraySize; i++ )
{
        if(  (i%16)  == 0  )
        {
                __builtin_prefetch ( &a[ i+PD ], WILL_READ_AND_WRITE, LOCALITY_LOW );
                __builtin_prefetch ( &b[ i+PD ], WILL_READ_ONLY,      LOCALITY_LOW );
        }
        a[ i ] = a[ i ] + b[ i ];
}
```

vprefetch0

vprefetch1

There can be two memory prefetches inside a loop

# Prefetching in Visual Studio



```
void _m_prefetch( void * );
```

Loads a cache line into cache and sets the cache line state to **exclusive**.

```
#define  PD          32                    // prefetch distance (fp words)

for( int i = 0; i < ArraySize; i++ )
{
        if(  (i%16)  == 0  )
        {
                _m_prefetch( &a[ i+PD ]  );
                _m_prefetch( &b[ i+PD ]  );
        }
        a[ i ] = a[ i ] + b[ i ];
}
```

## Array Multiplication Example

Length of Arrays (NUM): 1,000,000
Number of pairs of floats processed per SIMD call (SIMDSIZE): 4
Prefetch Distance (PD): 32

```
for(  int i = 0;  i < NUM;  i += SIMDSIZE)
{
        __builtin_prefetch ( &A[ i+PD ],  WILL_READ_ONLY,          LOCALITY_LOW );
        __builtin_prefetch ( &B[ i+PD ],  WILL_READ_ONLY,          LOCALITY_LOW );
        __builtin_prefetch ( &C[ i+PD ],  WILL_READ_AND_WRITE,  LOCALITY_LOW );

        SimdMul( &A[ i ], &B[ iI ],  &C[ i ],  SIMDSIZE);
}
```

# The Effects of Prefetching on SIMD Computations