Copy of compiler optimization reports with stream.c

Vendor compilers typically provide the most information about their optimizations of your code. Some may also provide an inline source listing (Cray and Intel below) where the optimization comments and labels appear next to the code.

Cray compiler (craycc: defaults -O3 and OpenMP enabled)

```
arnoldg@h2ologin4:~/stream> cc -c -hmsgs -hlist=ai -DTUNED stream.c
arnoldg@h2ologin4:~/stream> cat stream.lst
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                          S u m m a r y   R e p o r t
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Compilation
-----------
File     : /mnt/a/u/staff/arnoldg/stream/stream.c
Compiled : 2021-03-05  09:16:31
Compiler : Version 8.7.7
Ftnlx    : Version 8503 (libcif 85008)
Target   : x86-64
Command  : driver.cc -h cpu=interlagos -h static -D __CRAYXE
           -D __CRAY_INTERLAGOS -D __CRAYXT_COMPUTE_LINUX_TARGET
           -h network=gemini -c -h msgs -h list=ai -D TUNED stream.c
           -isystem /opt/cray/cce/8.7.7/cce/x86_64/include/craylibs
           -isystem /opt/cray/cce/8.7.7/cce/x86_64/include/basic
           -isystem /opt/gcc/6.1.0/snos/lib/gcc/x86_64-suse-linux/6.1.0/include
           -isystem /opt/gcc/6.1.0/snos/lib/gcc/x86_64-suse-linux/6.1.0/include-
           fixed -isystem /opt/gcc/6.1.0/snos/include -isystem /usr/include
           -I /opt/cray/mpt/7.7.4/gni/mpich-cray/8.6/include
           -I /opt/cray/libsci/18.12.1/CRAY/8.6/x86_64/include
           -I /opt/cray/rca/1.0.0-2.0502.60530.1.63.gem/include
           -I /opt/cray/alps/5.2.4-2.0502.9774.31.12.gem/include
           -I /opt/cray/xpmem/0.1-2.0502.64982.5.3.gem/include
           -I /opt/cray/gni-headers/4.0-1.0502.10859.7.8.gem/include
           -I /opt/cray/dmapp/7.0.1-1.0502.11080.8.74.gem/include
           -I /opt/cray/pmi/5.0.14/include
           -I /opt/cray/ugni/6.0-1.0502.10863.8.28.gem/include
           -I /opt/cray/udreg/2.3.2-1.0502.10518.2.17.gem/include
           -I /usr/local/include
           -I /opt/cray/wlm_detect/1.0-1.0502.64649.2.2.gem/include
           -I /opt/cray/krca/1.0.0-2.0502.63139.4.30.gem/include
           -I /opt/cray-hss-devel/7.2.0/include

clx report
------------
Source   : /mnt/a/u/staff/arnoldg/stream/stream.c
Date     : 03/05/2021  09:16:32


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                          O p t i o n s   R e p o r t
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Options :  -h cache2,scalar2,thread2,vector2,mpi0,ipa3,noaggress
           -h autoprefetch,noautothread,fusion2,msgs,nonegmsgs
           -h nooverindex,pattern,unroll2,nozeroinc
           -h noadd_paren,noupc,dwarf,fma,nofp_trap,nofunc_trace
           -h noomp_analyze,noomp_trace,nopat_trace
           -h omp,noacc
           -h c99,noexceptions,noconform,noinfinitevl
           -h notolerant,gnu
           -h safe_addr,thread_do_concurrent,fp2=approx,flex_mp=default
           -h alias=default:standard_restrict
           -h static (or -static)
           -h cpu=x86-64,interlagos
           -h network=gemini
           -K trap=none
```

%%
                    S o u r c e   L i s t i n g
%%

    %%%   L o o p m a r k   L e g e n d   %%%

    Primary Loop Type   Modifiers
    ------- ---- ----   ---------
    A - Pattern matched   a - atomic memory operation
                             b - blocked
    C - Collapsed   c - conditional and/or computed
    D - Deleted
    E - Cloned
    F - Flat - No calls   f - fused
    G - Accelerated   g - partitioned
    I - Inlined   i - interchanged
    M - Multithreaded   m - partitioned
                             n - non-blocking remote transfer
                             p - partial
    R - Rerolling   r - unrolled
                             s - shortloop
    V - Vectorized   w - unwound

    + - More messages listed at end of listing

```
  1.                    /*-----------------------------------------------------------------------*/
  2.                    /* Program: Stream                                                       */
  3.                    /* Revision: $Id: stream.c,v 5.9 2009/04/11 16:35:00 mccalpin Exp $ */
  4.                    /* Original code developed by John D. McCalpin                           */
  5.                    /* Programmers: John D. McCalpin                                         */
  6.                    /*              Joe R. Zagar                                             */
  7.                    /*                                                                       */
  8.                    /* This program measures memory transfer rates in MB/s for simple        */
  9.                    /* computational kernels coded in C.                                     */
 10.                    /*-----------------------------------------------------------------------*/
 11.                    /* Copyright 1991-2005: John D. McCalpin                                 */
 12.                    /*-----------------------------------------------------------------------*/
 13.                    /* License:                                                              */
 14.                    /*  1. You are free to use this program and/or to redistribute           */
 15.                    /*     this program.                                                     */
 16.                    /*  2. You are free to modify this program for your own use,             */
 17.                    /*     including commercial use, subject to the publication              */
 18.                    /*     restrictions in item 3.                                           */
 19.                    /*  3. You are free to publish results obtained from running this        */
 20.                    /*     program, or from works that you derive from this program,         */
 21.                    /*     with the following limitations:                                   */
 22.                    /*     3a. In order to be referred to as "STREAM benchmark results",     */
 23.                    /*         published results must be in conformance to the STREAM        */
 24.                    /*         Run Rules, (briefly reviewed below) published at              */
 25.                    /*         http://www.cs.virginia.edu/stream/ref.html                    */
 26.                    /*         and incorporated herein by reference.                         */
 27.                    /*         As the copyright holder, John McCalpin retains the            */
 28.                    /*         right to determine conformity with the Run Rules.             */
 29.                    /*     3b. Results based on modified source code or on runs not in       */
 30.                    /*         accordance with the STREAM Run Rules must be clearly          */
 31.                    /*         labelled whenever they are published.  Examples of            */
 32.                    /*         proper labelling include:                                     */
 33.                    /*         "tuned STREAM benchmark results"                              */
 34.                    /*         "based on a variant of the STREAM benchmark code"             */
 35.                    /*         Other comparable, clear and reasonable labelling is           */
 36.                    /*         acceptable.                                                   */
 37.                    /*     3c. Submission of results to the STREAM benchmark web site        */
 38.                    /*         is encouraged, but not required.                              */
 39.                    /*  4. Use of this program or creation of derived works based on this    */
 40.                    /*     program constitutes acceptance of these licensing restrictions.   */
 41.                    /*  5. Absolutely no warranty is expressed or implied.                   */
 42.                    /*-----------------------------------------------------------------------*/
 43.                    # include <stdio.h>
```

```
# include <math.h>
# include <float.h>
# include <limits.h>
# include <sys/time.h>

/* INSTRUCTIONS:
 *
 *  1) Stream requires a good bit of memory to run.  Adjust the
 *          value of 'N' (below) to give a 'timing calibration' of
 *          at least 20 clock-ticks.  This will provide rate estimates
 *          that should be good to about 5% precision.
 */

#ifndef N
#   define N        40000000
#endif
#ifndef NTIMES
#   define NTIMES   10
#endif
#ifndef OFFSET
#   define OFFSET   0
#endif

/*
 *  3) Compile the code with full optimization.  Many compilers
 *     generate unreasonably bad code before the optimizer tightens
 *     things up.  If the results are unreasonably good, on the
 *     other hand, the optimizer might be too smart for me!
 *
 *         Try compiling with:
 *               cc -O stream_omp.c -o stream_omp
 *
 *         This is known to work on Cray, SGI, IBM, and Sun machines.
 *
 *
 *  4) Mail the results to mccalpin@cs.virginia.edu
 *     Be sure to include:
 *          a) computer hardware model number and software revision
 *          b) the compiler flags
 *          c) all of the output from the test case.
 * Thanks!
 *
 */

# define HLINE "-------------------------------------------------------------\n"

# ifndef MIN
# define MIN(x,y) ((x)<(y)?(x):(y))
# endif
# ifndef MAX
# define MAX(x,y) ((x)>(y)?(x):(y))
# endif

static double       a[N+OFFSET],
            b[N+OFFSET],
            c[N+OFFSET];

static double       avgtime[4] = {0}, maxtime[4] = {0},
            mintime[4] = {FLT_MAX,FLT_MAX,FLT_MAX,FLT_MAX};

static char *label[4] = {"Copy:      ", "Scale:     ",
    "Add:       ", "Triad:     "};

static double       bytes[4] = {
    2 * sizeof(double) * N,
    2 * sizeof(double) * N,
    3 * sizeof(double) * N,
    3 * sizeof(double) * N
    };

extern double mysecond();
```

```
  115.                     extern void checkSTREAMresults();
  116.                     #ifdef TUNED
  117.                     extern void tuned_STREAM_Copy();
  118.                     extern void tuned_STREAM_Scale(double scalar);
  119.                     extern void tuned_STREAM_Add();
  120.                     extern void tuned_STREAM_Triad(double scalar);
  121.                     #endif
  122.                     #ifdef _OPENMP
  123.                     extern int omp_get_num_threads();
  124.                     #endif
  125.                     int
  126.                     main()
  127.                         {
  128.                         int                     quantum, checktick();
  129.                         int                     BytesPerWord;
  130.                         register int    j, k;
  131.                         double          scalar, t, times[4][NTIMES];
  132.
  133.                         /* --- SETUP --- determine precision and check timing --- */
  134.
  135.  +                      printf(HLINE);
                              ^
CC-3021 CC: IPA main, File = stream.c, Line = 135, Column = 5
  "printf" (called from "main") was not inlined because the compiler was unable to locate the routine.

  136.  +                      printf("STREAM version $Revision: 5.9 $\n");
                              ^
CC-3021 CC: IPA main, File = stream.c, Line = 136, Column = 5
  "printf" (called from "main") was not inlined because the compiler was unable to locate the routine.

  137.  +                      printf(HLINE);
                              ^
CC-3021 CC: IPA main, File = stream.c, Line = 137, Column = 5
  "printf" (called from "main") was not inlined because the compiler was unable to locate the routine.

  138.                         BytesPerWord = sizeof(double);
  139.  +                      printf("This system uses %d bytes per DOUBLE PRECISION word.\n",
                              ^
CC-3021 CC: IPA main, File = stream.c, Line = 139, Column = 5
  "printf" (called from "main") was not inlined because the compiler was unable to locate the routine.

  140.                         BytesPerWord);
  141.
  142.  +                      printf(HLINE);
                              ^
CC-3021 CC: IPA main, File = stream.c, Line = 142, Column = 5
  "printf" (called from "main") was not inlined because the compiler was unable to locate the routine.

  143.                     #ifdef NO_LONG_LONG
  144.                         printf("Array size = %d, Offset = %d\n" , N, OFFSET);
  145.                     #else
  146.  +                      printf("Array size = %llu, Offset = %d\n", (unsigned long long) N, OFFSET);
                              ^
CC-3021 CC: IPA main, File = stream.c, Line = 146, Column = 5
  "printf" (called from "main") was not inlined because the compiler was unable to locate the routine.

  147.                     #endif
  148.
  149.  +                      printf("Total memory required = %.1f MB.\n",
                              ^
CC-3021 CC: IPA main, File = stream.c, Line = 149, Column = 5
  "printf" (called from "main") was not inlined because the compiler was unable to locate the routine.

  150.                         (3.0 * BytesPerWord) * ( (double) N / 1048576.0));
  151.  +                      printf("Each test is run %d times, but only\n", NTIMES);
                              ^
CC-3021 CC: IPA main, File = stream.c, Line = 151, Column = 5
  "printf" (called from "main") was not inlined because the compiler was unable to locate the routine.

  152.  +                      printf("the *best* time for each is used.\n");
                              ^
```

```
CC-3021 CC: IPA main, File = stream.c, Line = 152, Column = 5
  "printf" (called from "main") was not inlined because the compiler was unable to locate the routine.

  153.
  154.                      #ifdef _OPENMP
  155.  +                       printf(HLINE);
                                ^
CC-3021 CC: IPA main, File = stream.c, Line = 155, Column = 5
  "printf" (called from "main") was not inlined because the compiler was unable to locate the routine.

  156.                      #pragma omp parallel
  157.       M------------<     {
CC-6823 CC: THREAD main, File = stream.c, Line = 157
  A region starting at line 157 and ending at line 163 was multi-threaded.

  158.       M              #pragma omp master
  159.       M                  {
  160.  +    M                      k = omp_get_num_threads();
                                  ^
CC-3021 CC: IPA main, File = stream.c, Line = 160, Column = 6
  "omp_get_num_threads" (called from "main") was not inlined because the compiler was unable to locate the
routine.

  161.  +    M                      printf ("Number of Threads requested = %i\n",k);
                                  ^
CC-3021 CC: IPA main, File = stream.c, Line = 161, Column = 6
  "printf" (called from "main") was not inlined because the compiler was unable to locate the routine.

  162.       M                      }
  163.       M------------>     }
  164.                      #endif
  165.
  166.  +                       printf(HLINE);
                                ^
CC-3021 CC: IPA main, File = stream.c, Line = 166, Column = 5
  "printf" (called from "main") was not inlined because the compiler was unable to locate the routine.

  167.                      #pragma omp parallel
  168.  + M-<                   {
CC-6831 CC: THREAD main, File = stream.c, Line = 168
  An expanded multi-threaded region was created starting near line 168 and ending near line 178.

CC-6824 CC: THREAD main, File = stream.c, Line = 168
  A region starting at line 168 and ending at line 170 was multi-threaded and merged with an expanded multi-
thread region.

  169.  + M                    printf ("Printing one line per active thread....\n");
                                ^
CC-3021 CC: IPA main, File = stream.c, Line = 169, Column = 5
  "printf" (called from "main") was not inlined because the compiler was unable to locate the routine.

  170.    M                    }
  171.    M
  172.    M                    /* Get initial value for system clock. */
  173.    M                #pragma omp parallel for
  174.  + M  mA------------<     for (j=0; j<N; j++) {
CC-6230 CC: VECTOR main, File = stream.c, Line = 174
  A loop was replaced with multiple library calls.

CC-6824 CC: THREAD main, File = stream.c, Line = 174
  A region starting at line 174 and ending at line 178 was multi-threaded and merged with an expanded multi-
thread region.

CC-6817 CC: THREAD main, File = stream.c, Line = 174
  A loop was partitioned.

  175.    M  mA                a[j] = 1.0;
  176.    M  mA                b[j] = 2.0;
  177.    M  mA                c[j] = 0.0;
  178.    M->mA------------>     }
  179.
```

```
  180.  +                       printf(HLINE);
                                ^
CC-3021 CC: IPA main, File = stream.c, Line = 180, Column = 5
  "printf" (called from "main") was not inlined because the compiler was unable to locate the routine.

  181.
  182.  +                       if  ( (quantum = checktick()) >= 1)
                                ^
CC-3118 CC: IPA main, File = stream.c, Line = 182, Column = 5
  "checktick" (called from "main") was not inlined because the call site will not flatten.  "gettimeofday" is
missing.

  183.  +                       printf("Your clock granularity/precision appears to be "
                                ^
CC-3021 CC: IPA main, File = stream.c, Line = 183, Column = 2
  "printf" (called from "main") was not inlined because the compiler was unable to locate the routine.

  184.                            "%d microseconds.\n", quantum);
  185.                          else {
  186.  +                       printf("Your clock granularity appears to be "
                                ^
CC-3021 CC: IPA main, File = stream.c, Line = 186, Column = 2
  "printf" (called from "main") was not inlined because the compiler was unable to locate the routine.

  187.                            "less than one microsecond.\n");
  188.                          quantum = 1;
  189.                          }
  190.
  191.  +                       t = mysecond();
                                ^
CC-3118 CC: IPA main, File = stream.c, Line = 191, Column = 5
  "mysecond" (called from "main") was not inlined because the call site will not flatten.  "gettimeofday" is
missing.

  192.                      #pragma omp parallel for
  193.       MmVr4--------<     for (j = 0; j < N; j++)
CC-6005 CC: SCALAR main, File = stream.c, Line = 193
  A loop was unrolled 4 times.

CC-6823 CC: THREAD main, File = stream.c, Line = 193
  A region starting at line 193 and ending at line 194 was multi-threaded.

CC-6204 CC: VECTOR main, File = stream.c, Line = 193
  A loop was vectorized.

CC-6817 CC: THREAD main, File = stream.c, Line = 193
  A loop was partitioned.

  194.       MmVr4-------->     a[j] = 2.0E0 * a[j];
  195.  +                       t = 1.0E6 * (mysecond() - t);
                                ^
CC-3118 CC: IPA main, File = stream.c, Line = 195, Column = 5
  "mysecond" (called from "main") was not inlined because the call site will not flatten.  "gettimeofday" is
missing.

  196.
  197.  +                       printf("Each test below will take on the order"
                                ^
CC-3021 CC: IPA main, File = stream.c, Line = 197, Column = 5
  "printf" (called from "main") was not inlined because the compiler was unable to locate the routine.

  198.                          " of %d microseconds.\n", (int) t  );
  199.  +                       printf("   (= %d clock ticks)\n", (int) (t/quantum) );
                                ^
CC-3021 CC: IPA main, File = stream.c, Line = 199, Column = 5
  "printf" (called from "main") was not inlined because the compiler was unable to locate the routine.

  200.  +                       printf("Increase the size of the arrays if this shows that\n");
                                ^
CC-3021 CC: IPA main, File = stream.c, Line = 200, Column = 5
  "printf" (called from "main") was not inlined because the compiler was unable to locate the routine.
```

```
  201.  +                       printf("you are not getting at least 20 clock ticks per test.\n");
                                ^
CC-3021 CC: IPA main, File = stream.c, Line = 201, Column = 5
  "printf" (called from "main") was not inlined because the compiler was unable to locate the routine.

  202.
  203.  +                       printf(HLINE);
                                ^
CC-3021 CC: IPA main, File = stream.c, Line = 203, Column = 5
  "printf" (called from "main") was not inlined because the compiler was unable to locate the routine.

  204.
  205.  +                       printf("WARNING -- The above is only a rough guideline.\n");
                                ^
CC-3021 CC: IPA main, File = stream.c, Line = 205, Column = 5
  "printf" (called from "main") was not inlined because the compiler was unable to locate the routine.

  206.  +                       printf("For best results, please be sure you know the\n");
                                ^
CC-3021 CC: IPA main, File = stream.c, Line = 206, Column = 5
  "printf" (called from "main") was not inlined because the compiler was unable to locate the routine.

  207.  +                       printf("precision of your system timer.\n");
                                ^
CC-3021 CC: IPA main, File = stream.c, Line = 207, Column = 5
  "printf" (called from "main") was not inlined because the compiler was unable to locate the routine.

  208.  +                       printf(HLINE);
                                ^
CC-3021 CC: IPA main, File = stream.c, Line = 208, Column = 5
  "printf" (called from "main") was not inlined because the compiler was unable to locate the routine.

  209.
  210.                          /*      --- MAIN LOOP --- repeat test cases NTIMES times --- */
  211.
  212.                          scalar = 3.0;
  213.  +    1------------<     for (k=0; k<NTIMES; k++)
CC-6287 CC: VECTOR main, File = stream.c, Line = 213
  A loop was not vectorized because it contains a call to function "mysecond" on line 215.

  214.       1                 {
  215.  +    1                 times[0][k] = mysecond();
                                ^
CC-3118 CC: IPA main, File = stream.c, Line = 215, Column = 2
  "mysecond" (called from "main") was not inlined because the call site will not flatten.  "gettimeofday" is
missing.

  216.       1              #ifdef TUNED
  217.       1 MmA I-----<>         tuned_STREAM_Copy();
CC-6202 CC: VECTOR main, File = stream.c, Line = 217
  A loop was replaced by a library call.

CC-6823 CC: THREAD main, File = stream.c, Line = 217
  A region starting at line 217 and ending at line 217 was multi-threaded.

CC-6817 CC: THREAD main, File = stream.c, Line = 217
  A loop was partitioned.

                                  ^
CC-3001 CC: IPA main, File = stream.c, Line = 217, Column = 9
  The call to tiny leaf routine "tuned_STREAM_Copy" was textually inlined.

  218.       1              #else
  219.       1              #pragma omp parallel for
  220.       1                  for (j=0; j<N; j++)
  221.       1                      c[j] = a[j];
  222.       1              #endif
  223.  +    1                  times[0][k] = mysecond() - times[0][k];
                                ^
CC-3118 CC: IPA main, File = stream.c, Line = 223, Column = 2
```

```
  "mysecond" (called from "main") was not inlined because the call site will not flatten.  "gettimeofday" is
missing.

  224.       1
  225.  +    1                  times[1][k] = mysecond();
                                ^
CC-3118 CC: IPA main, File = stream.c, Line = 225, Column = 2
  "mysecond" (called from "main") was not inlined because the call site will not flatten.  "gettimeofday" is
missing.

  226.       1              #ifdef TUNED
  227.       1 MmVr4 I---<>         tuned_STREAM_Scale(scalar);
CC-6005 CC: SCALAR main, File = stream.c, Line = 227
  A loop was unrolled 4 times.

CC-6823 CC: THREAD main, File = stream.c, Line = 227
  A region starting at line 227 and ending at line 227 was multi-threaded.

CC-6204 CC: VECTOR main, File = stream.c, Line = 227
  A loop was vectorized.

CC-6817 CC: THREAD main, File = stream.c, Line = 227
  A loop was partitioned.

                                    ^
CC-3001 CC: IPA main, File = stream.c, Line = 227, Column = 9
  The call to tiny leaf routine "tuned_STREAM_Scale" was textually inlined.

  228.       1              #else
  229.       1              #pragma omp parallel for
  230.       1                  for (j=0; j<N; j++)
  231.       1                      b[j] = scalar*c[j];
  232.       1              #endif
  233.  +    1                  times[1][k] = mysecond() - times[1][k];
                                ^
CC-3118 CC: IPA main, File = stream.c, Line = 233, Column = 2
  "mysecond" (called from "main") was not inlined because the call site will not flatten.  "gettimeofday" is
missing.

  234.       1
  235.  +    1                  times[2][k] = mysecond();
                                ^
CC-3118 CC: IPA main, File = stream.c, Line = 235, Column = 2
  "mysecond" (called from "main") was not inlined because the call site will not flatten.  "gettimeofday" is
missing.

  236.       1              #ifdef TUNED
  237.       1 MmVr4 I---<>         tuned_STREAM_Add();
CC-6005 CC: SCALAR main, File = stream.c, Line = 237
  A loop was unrolled 4 times.

CC-6823 CC: THREAD main, File = stream.c, Line = 237
  A region starting at line 237 and ending at line 237 was multi-threaded.

CC-6204 CC: VECTOR main, File = stream.c, Line = 237
  A loop was vectorized.

CC-6817 CC: THREAD main, File = stream.c, Line = 237
  A loop was partitioned.

                                    ^
CC-3001 CC: IPA main, File = stream.c, Line = 237, Column = 9
  The call to tiny leaf routine "tuned_STREAM_Add" was textually inlined.

  238.       1              #else
  239.       1              #pragma omp parallel for
  240.       1                  for (j=0; j<N; j++)
  241.       1                      c[j] = a[j]+b[j];
  242.       1              #endif
  243.  +    1                  times[2][k] = mysecond() - times[2][k];
                                ^
```

```
CC-3118 CC: IPA main, File = stream.c, Line = 243, Column = 2
  "mysecond" (called from "main") was not inlined because the call site will not flatten.  "gettimeofday" is
missing.

  244.       1
  245.  +    1                  times[3][k] = mysecond();
                                 ^
CC-3118 CC: IPA main, File = stream.c, Line = 245, Column = 2
  "mysecond" (called from "main") was not inlined because the call site will not flatten.  "gettimeofday" is
missing.

  246.       1              #ifdef TUNED
  247.       1 MmVr4 I---<>         tuned_STREAM_Triad(scalar);
CC-6005 CC: SCALAR main, File = stream.c, Line = 247
  A loop was unrolled 4 times.

CC-6823 CC: THREAD main, File = stream.c, Line = 247
  A region starting at line 247 and ending at line 247 was multi-threaded.

CC-6204 CC: VECTOR main, File = stream.c, Line = 247
  A loop was vectorized.

CC-6817 CC: THREAD main, File = stream.c, Line = 247
  A loop was partitioned.

                                    ^
CC-3001 CC: IPA main, File = stream.c, Line = 247, Column = 9
  The call to tiny leaf routine "tuned_STREAM_Triad" was textually inlined.

  248.       1              #else
  249.       1              #pragma omp parallel for
  250.       1                  for (j=0; j<N; j++)
  251.       1                      a[j] = b[j]+scalar*c[j];
  252.       1              #endif
  253.  +    1                  times[3][k] = mysecond() - times[3][k];
                                 ^
CC-3118 CC: IPA main, File = stream.c, Line = 253, Column = 2
  "mysecond" (called from "main") was not inlined because the call site will not flatten.  "gettimeofday" is
missing.

  254.       1------------>     }
  255.
  256.                         /*      --- SUMMARY --- */
  257.
  258.  +    iVw----------<     for (k=1; k<NTIMES; k++) /* note -- skip first iteration */
CC-6007 CC: SCALAR main, File = stream.c, Line = 258
  A loop was interchanged with the loop starting at line 260.

CC-6373 CC: VECTOR main, File = stream.c, Line = 258
  A loop with a trip count of 9 was unwound into 2 vector iterations.

CC-6382 CC: VECTOR main, File = stream.c, Line = 258
  A loop was partially vector pipelined.

CC-6204 CC: VECTOR main, File = stream.c, Line = 258
  A loop was vectorized.

  259.       iVw                {
  260.  +    iVw i--------<     for (j=0; j<4; j++)
CC-6294 CC: VECTOR main, File = stream.c, Line = 260
  A loop was not vectorized because a better candidate was found at line 258.

  261.       iVw i                  {
  262.       iVw i                  avgtime[j] = avgtime[j] + times[j][k];
  263.       iVw i                  mintime[j] = MIN(mintime[j], times[j][k]);
  264.       iVw i                  maxtime[j] = MAX(maxtime[j], times[j][k]);
  265.       iVw i-------->         }
  266.       iVw---------->     }
  267.
  268.  +                       printf("Function      Rate (MB/s)   Avg time     Min time     Max time\n");
                                 ^
```

```
CC-3021 CC: IPA main, File = stream.c, Line = 268, Column = 5
  "printf" (called from "main") was not inlined because the compiler was unable to locate the routine.

  269.  +    1------------<     for (j=0; j<4; j++) {
CC-6287 CC: VECTOR main, File = stream.c, Line = 269
  A loop was not vectorized because it contains a call to function "printf" on line 272.

  270.       1                 avgtime[j] = avgtime[j]/(double)(NTIMES-1);
  271.       1
  272.  +    1                 printf("%s%11.4f  %11.4f  %11.4f  %11.4f\n", label[j],
                                 ^
CC-3021 CC: IPA main, File = stream.c, Line = 272, Column = 2
  "printf" (called from "main") was not inlined because the compiler was unable to locate the routine.

  273.       1                       1.0E-06 * bytes[j]/mintime[j],
  274.       1                       avgtime[j],
  275.       1                       mintime[j],
  276.       1                       maxtime[j]);
  277.       1------------>     }
  278.  +                       printf(HLINE);
                                 ^
CC-3021 CC: IPA main, File = stream.c, Line = 278, Column = 5
  "printf" (called from "main") was not inlined because the compiler was unable to locate the routine.

  279.
  280.                         /* --- Check Results --- */
  281.  +                      checkSTREAMresults();
                                 ^
CC-3118 CC: IPA main, File = stream.c, Line = 281, Column = 5
  "checkSTREAMresults" (called from "main") was not inlined because the call site will not flatten.  "printf"
is missing.

  282.  +                      printf(HLINE);
                                 ^
CC-3021 CC: IPA main, File = stream.c, Line = 282, Column = 5
  "printf" (called from "main") was not inlined because the compiler was unable to locate the routine.

  283.
  284.                         return 0;
  285.                     }
  286.
  287.                     # define    M       20
  288.
  289.                     int
  290.                     checktick()
  291.                         {
  292.                         int             i, minDelta, Delta;
  293.                         double  t1, t2, timesfound[M];
  294.
  295.                     /*  Collect a sequence of M unique time values from the system. */
  296.
  297.  +    1------------<     for (i = 0; i < M; i++) {
CC-6287 CC: VECTOR checktick, File = stream.c, Line = 297
  A loop was not vectorized because it contains a call to function "mysecond" on line 298.

  298.  +    1                 t1 = mysecond();
                                 ^
CC-3118 CC: IPA checktick, File = stream.c, Line = 298, Column = 2
  "mysecond" (called from "checktick") was not inlined because the call site will not flatten.  "gettimeofday"
is missing.

  299.  +    1 2----------<     while( ((t2=mysecond()) - t1) < 1.0E-6 )
CC-6287 CC: VECTOR checktick, File = stream.c, Line = 299
  A loop was not vectorized because it contains a call to function "mysecond" on line 299.

                                 ^
CC-3118 CC: IPA checktick, File = stream.c, Line = 299, Column = 2
  "mysecond" (called from "checktick") was not inlined because the call site will not flatten.  "gettimeofday"
is missing.

                                 ^
```

```
CC-3118 CC: IPA checktick, File = stream.c, Line = 299, Column = 2
  "mysecond" (called from "checktick") was not inlined because the call site will not flatten.  "gettimeofday"
is missing.

  300.       1 2---------->         ;
  301.       1                   timesfound[i] = t1 = t2;
  302.       1------------>      }
  303.
  304.                       /*
  305.                        * Determine the minimum difference between these M values.
  306.                        * This result will be our estimate (in microseconds) for the
  307.                        * clock granularity.
  308.                        */
  309.
  310.                           minDelta = 1000000;
  311.  +    Vw-----------<      for (i = 1; i < M; i++) {
CC-6373 CC: VECTOR checktick, File = stream.c, Line = 311
  A loop with a trip count of 19 was unwound into 4 vector iterations.

CC-6382 CC: VECTOR checktick, File = stream.c, Line = 311
  A loop was partially vector pipelined.

CC-6204 CC: VECTOR checktick, File = stream.c, Line = 311
  A loop was vectorized.

  312.       Vw                  Delta = (int)( 1.0E6 * (timesfound[i]-timesfound[i-1]));
  313.       Vw                  minDelta = MIN(minDelta, MAX(Delta,0));
  314.       Vw----------->      }
  315.
  316.                          return(minDelta);
  317.                           }
  318.
  319.
  320.
  321.                       /* A gettimeofday routine to give access to the wall
  322.                          clock timer on most UNIX-like systems.  */
  323.
  324.                       #include <sys/time.h>
  325.
  326.                       double mysecond()
  327.                       {
  328.                               struct timeval tp;
  329.                               struct timezone tzp;
  330.                               int i;
  331.
  332.  +                            i = gettimeofday(&tp,&tzp);
                                    ^
CC-3021 CC: IPA mysecond, File = stream.c, Line = 332, Column = 9
  "gettimeofday" (called from "mysecond") was not inlined because the compiler was unable to locate the routine.

  333.                               return ( (double) tp.tv_sec + (double) tp.tv_usec * 1.e-6 );
  334.                       }
  335.
  336.                       void checkSTREAMresults ()
  337.                       {
  338.                           double aj,bj,cj,scalar;
  339.                           double asum,bsum,csum;
  340.                           double epsilon;
  341.                           int     j,k;
  342.
  343.                           /* reproduce initialization */
  344.                           aj = 1.0;
  345.                           bj = 2.0;
  346.                           cj = 0.0;
  347.                           /* a[] is modified during timing check */
  348.                           aj = 2.0E0 * aj;
  349.                           /* now execute timing loop */
  350.                           scalar = 3.0;
  351.       V------------<      for (k=0; k<NTIMES; k++)
CC-6204 CC: VECTOR checkSTREAMresults, File = stream.c, Line = 351
  A loop was vectorized.
```

```
  352.       V                       {
  353.       V                           cj = aj;
  354.       V                           bj = scalar*cj;
  355.       V                           cj = aj+bj;
  356.       V                           aj = bj+scalar*cj;
  357.       V------------>          }
  358.                           aj = aj * (double) (N);
  359.                           bj = bj * (double) (N);
  360.                           cj = cj * (double) (N);
  361.
  362.                           asum = 0.0;
  363.                           bsum = 0.0;
  364.                           csum = 0.0;
  365.       Vr4---------<       for (j=0; j<N; j++) {
CC-6005 CC: SCALAR checkSTREAMresults, File = stream.c, Line = 365
  A loop was unrolled 4 times.

CC-6204 CC: VECTOR checkSTREAMresults, File = stream.c, Line = 365
  A loop was vectorized.

  366.       Vr4                           asum += a[j];
  367.       Vr4                           bsum += b[j];
  368.       Vr4                           csum += c[j];
  369.       Vr4---------->      }
  370.                         #ifdef VERBOSE
  371.                           printf ("Results Comparison: \n");
  372.                           printf ("        Expected  : %f %f %f \n",aj,bj,cj);
  373.                           printf ("        Observed  : %f %f %f \n",asum,bsum,csum);
  374.                         #endif
  375.
  376.                         #ifndef abs
  377.                         #define abs(a) ((a) >= 0 ? (a) : -(a))
  378.                         #endif
  379.                           epsilon = 1.e-8;
  380.
  381.                           if (abs(aj-asum)/asum > epsilon) {
  382.  +                                printf ("Failed Validation on array a[]\n");
                                   ^
CC-3021 CC: IPA checkSTREAMresults, File = stream.c, Line = 382, Column = 3
  "printf" (called from "checkSTREAMresults") was not inlined because the compiler was unable to locate the
routine.

  383.  +                                printf ("        Expected  : %f \n",aj);
                                   ^
CC-3021 CC: IPA checkSTREAMresults, File = stream.c, Line = 383, Column = 3
  "printf" (called from "checkSTREAMresults") was not inlined because the compiler was unable to locate the
routine.

  384.  +                                printf ("        Observed  : %f \n",asum);
                                   ^
CC-3021 CC: IPA checkSTREAMresults, File = stream.c, Line = 384, Column = 3
  "printf" (called from "checkSTREAMresults") was not inlined because the compiler was unable to locate the
routine.

  385.                           }
  386.                           else if (abs(bj-bsum)/bsum > epsilon) {
  387.  +                                printf ("Failed Validation on array b[]\n");
                                   ^
CC-3021 CC: IPA checkSTREAMresults, File = stream.c, Line = 387, Column = 3
  "printf" (called from "checkSTREAMresults") was not inlined because the compiler was unable to locate the
routine.

  388.  +                                printf ("        Expected  : %f \n",bj);
                                   ^
CC-3021 CC: IPA checkSTREAMresults, File = stream.c, Line = 388, Column = 3
  "printf" (called from "checkSTREAMresults") was not inlined because the compiler was unable to locate the
routine.

  389.  +                                printf ("        Observed  : %f \n",bsum);
                                   ^
```

```
CC-3021 CC: IPA checkSTREAMresults, File = stream.c, Line = 389, Column = 3
  "printf" (called from "checkSTREAMresults") was not inlined because the compiler was unable to locate the
routine.

  390.                          }
  391.                          else if (abs(cj-csum)/csum > epsilon) {
  392.  +                               printf ("Failed Validation on array c[]\n");
                                        ^
CC-3021 CC: IPA checkSTREAMresults, File = stream.c, Line = 392, Column = 3
  "printf" (called from "checkSTREAMresults") was not inlined because the compiler was unable to locate the
routine.

  393.  +                               printf ("        Expected  : %f \n",cj);
                                        ^
CC-3021 CC: IPA checkSTREAMresults, File = stream.c, Line = 393, Column = 3
  "printf" (called from "checkSTREAMresults") was not inlined because the compiler was unable to locate the
routine.

  394.  +                               printf ("        Observed  : %f \n",csum);
                                        ^
CC-3021 CC: IPA checkSTREAMresults, File = stream.c, Line = 394, Column = 3
  "printf" (called from "checkSTREAMresults") was not inlined because the compiler was unable to locate the
routine.

  395.                          }
  396.                          else {
  397.  +                               printf ("Solution Validates\n");
                                        ^
CC-3021 CC: IPA checkSTREAMresults, File = stream.c, Line = 397, Column = 3
  "printf" (called from "checkSTREAMresults") was not inlined because the compiler was unable to locate the
routine.

  398.                          }
  399.                      }
  400.
  401.                      void tuned_STREAM_Copy()
  402.                      {
  403.                          int j;
  404.                      #pragma omp parallel for
  405.       MmA----------<         for (j=0; j<N; j++)
CC-6202 CC: VECTOR tuned_STREAM_Copy, File = stream.c, Line = 405
  A loop was replaced by a library call.

CC-6823 CC: THREAD tuned_STREAM_Copy, File = stream.c, Line = 405
  A region starting at line 405 and ending at line 406 was multi-threaded.

CC-6817 CC: THREAD tuned_STREAM_Copy, File = stream.c, Line = 405
  A loop was partitioned.

  406.       MmA---------->             c[j] = a[j];
  407.                      }
  408.
  409.                      void tuned_STREAM_Scale(double scalar)
  410.                      {
  411.                          int j;
  412.                      #pragma omp parallel for
  413.       MmVr4--------<     for (j=0; j<N; j++)
CC-6005 CC: SCALAR tuned_STREAM_Scale, File = stream.c, Line = 413
  A loop was unrolled 4 times.

CC-6823 CC: THREAD tuned_STREAM_Scale, File = stream.c, Line = 413
  A region starting at line 413 and ending at line 414 was multi-threaded.

CC-6204 CC: VECTOR tuned_STREAM_Scale, File = stream.c, Line = 413
  A loop was vectorized.

CC-6817 CC: THREAD tuned_STREAM_Scale, File = stream.c, Line = 413
  A loop was partitioned.

  414.       MmVr4-------->         b[j] = scalar*c[j];
  415.                      }
```

```
  416.
  417.                        void tuned_STREAM_Add()
  418.                        {
  419.                            int j;
  420.                        #pragma omp parallel for
  421.       MmVr4--------<      for (j=0; j<N; j++)
CC-6005 CC: SCALAR tuned_STREAM_Add, File = stream.c, Line = 421
  A loop was unrolled 4 times.

CC-6823 CC: THREAD tuned_STREAM_Add, File = stream.c, Line = 421
  A region starting at line 421 and ending at line 422 was multi-threaded.

CC-6204 CC: VECTOR tuned_STREAM_Add, File = stream.c, Line = 421
  A loop was vectorized.

CC-6817 CC: THREAD tuned_STREAM_Add, File = stream.c, Line = 421
  A loop was partitioned.

  422.       MmVr4-------->          c[j] = a[j]+b[j];
  423.                        }
  424.
  425.                        void tuned_STREAM_Triad(double scalar)
  426.                        {
  427.                            int j;
  428.                        #pragma omp parallel for
  429.       MmVr4--------<      for (j=0; j<N; j++)
CC-6005 CC: SCALAR tuned_STREAM_Triad, File = stream.c, Line = 429
  A loop was unrolled 4 times.

CC-6823 CC: THREAD tuned_STREAM_Triad, File = stream.c, Line = 429
  A region starting at line 429 and ending at line 430 was multi-threaded.

CC-6204 CC: VECTOR tuned_STREAM_Triad, File = stream.c, Line = 429
  A loop was vectorized.

CC-6817 CC: THREAD tuned_STREAM_Triad, File = stream.c, Line = 429
  A loop was partitioned.

  430.       MmVr4-------->          a[j] = b[j]+scalar*c[j];
  431.                        }
  432.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
arnoldg@h2ologin4:~/stream>
```

PGI compiler (pgcc)

```
arnoldg@h2ologin4:~/stream> cc -O3 -Minfo=all -mp -c -DTUNED stream.c
main:
    157, Parallel region activated
    158, Begin master region
    163, End master region
    166, Parallel region terminated
    168, Parallel region activated
    173, Parallel region terminated
    174, Parallel region activated
         Parallel loop activated with static block schedule
         Generated an alternate version of the loop
         Generated vector simd code for the loop
    180, Barrier
         Parallel region terminated
    182, checktick inlined, size=24 (inline) file stream.c (291)
         297, FMA (fused multiply-add) instruction(s) generated
         298, mysecond inlined, size=4 (inline) file stream.c (327)
         299, Loop not vectorized/parallelized: contains call
              FMA (fused multiply-add) instruction(s) generated
    191, mysecond inlined, size=4 (inline) file stream.c (327)
    191, FMA (fused multiply-add) instruction(s) generated
```

```
    193, Parallel region activated
         Parallel loop activated with static block schedule
         Generated vector simd code for the loop
         Generated a prefetch instruction for the loop
    195, mysecond inlined, size=4 (inline) file stream.c (327)
    195, Barrier
         Parallel region terminated
         FMA (fused multiply-add) instruction(s) generated
    213, Loop not vectorized/parallelized: contains call
         FMA (fused multiply-add) instruction(s) generated
    215, mysecond inlined, size=4 (inline) file stream.c (327)
    223, mysecond inlined, size=4 (inline) file stream.c (327)
    225, mysecond inlined, size=4 (inline) file stream.c (327)
    233, mysecond inlined, size=4 (inline) file stream.c (327)
    235, mysecond inlined, size=4 (inline) file stream.c (327)
    243, mysecond inlined, size=4 (inline) file stream.c (327)
    245, mysecond inlined, size=4 (inline) file stream.c (327)
    253, mysecond inlined, size=4 (inline) file stream.c (327)
    258, Loop not vectorized: data dependency
    260, Loop unrolled 4 times (completely unrolled)
    269, Loop not vectorized/parallelized: contains call
    281, checkSTREAMresults inlined, size=34 (inline) file stream.c (337)
         351, Loop unrolled 4 times
              FMA (fused multiply-add) instruction(s) generated
         365, Generated vector simd code for the loop containing reductions
              Generated 3 prefetch instructions for the loop
checktick:
    297, FMA (fused multiply-add) instruction(s) generated
    298, mysecond inlined, size=4 (inline) file stream.c (327)
    299, mysecond inlined, size=4 (inline) file stream.c (327)
    299, Loop not vectorized/parallelized: contains call
         FMA (fused multiply-add) instruction(s) generated
mysecond:
    332, FMA (fused multiply-add) instruction(s) generated
checkSTREAMresults:
    351, Loop unrolled 4 times
         FMA (fused multiply-add) instruction(s) generated
    365, Generated vector simd code for the loop containing reductions
         Generated 3 prefetch instructions for the loop
tuned_STREAM_Copy:
    405, Parallel region activated
         Parallel loop activated with static block schedule
         Memory copy idiom, loop replaced by call to __c_mcopy8
    407, Barrier
         Parallel region terminated
tuned_STREAM_Scale:
    413, Parallel region activated
         Parallel loop activated with static block schedule
         Generated an alternate version of the loop
         Generated vector simd code for the loop
         Generated a prefetch instruction for the loop
         Generated vector simd code for the loop
         Generated a prefetch instruction for the loop
    415, Barrier
         Parallel region terminated
tuned_STREAM_Add:
    421, Parallel region activated
         Parallel loop activated with static block schedule
         Generated an alternate version of the loop
         Generated vector simd code for the loop
         Generated 2 prefetch instructions for the loop
         Generated vector simd code for the loop
         Generated 2 prefetch instructions for the loop
    423, Barrier
         Parallel region terminated
tuned_STREAM_Triad:
    429, Parallel region activated
         Parallel loop activated with static block schedule
         Generated an alternate version of the loop
         Generated vector simd code for the loop
         Generated 2 prefetch instructions for the loop
```

```
         Generated vector simd code for the loop
         Generated 2 prefetch instructions for the loop
         FMA (fused multiply-add) instruction(s) generated
    431, Barrier
         Parallel region terminated
arnoldg@h2ologin4:~/stream>
```

Intel compiler (icc)

```
arnoldg@h2ologin4:~/stream> cc -c -qopt-report-annotate -DTUNED -O3 -qopenmp stream.c
arnoldg@h2ologin4:~/stream> cat stream.c.annot
//
// ------- Annotated listing with optimization reports for "/mnt/a/u/staff/arnoldg/stream/stream.c" -------
//
//INLINING OPTION VALUES:
//  -inline-factor: 100
//  -inline-min-size: 30
//  -inline-max-size: 230
//  -inline-max-total-size: 2000
//  -inline-max-per-routine: 10000
//  -inline-max-per-compile: 500000
//
1       /*-----------------------------------------------------------------------*/
2       /* Program: Stream                                                       */
3       /* Revision: $Id: stream.c,v 5.9 2009/04/11 16:35:00 mccalpin Exp $ */
4       /* Original code developed by John D. McCalpin                           */
5       /* Programmers: John D. McCalpin                                         */
6       /*              Joe R. Zagar                                             */
7       /*                                                                       */
8       /* This program measures memory transfer rates in MB/s for simple        */
9       /* computational kernels coded in C.                                     */
10      /*-----------------------------------------------------------------------*/
11      /* Copyright 1991-2005: John D. McCalpin                                 */
12      /*-----------------------------------------------------------------------*/
13      /* License:                                                              */
14      /*  1. You are free to use this program and/or to redistribute           */
15      /*     this program.                                                     */
16      /*  2. You are free to modify this program for your own use,             */
17      /*     including commercial use, subject to the publication             */
18      /*     restrictions in item 3.                                           */
19      /*  3. You are free to publish results obtained from running this        */
20      /*     program, or from works that you derive from this program,         */
21      /*     with the following limitations:                                   */
22      /*     3a. In order to be referred to as "STREAM benchmark results",     */
23      /*         published results must be in conformance to the STREAM        */
24      /*         Run Rules, (briefly reviewed below) published at              */
25      /*         http://www.cs.virginia.edu/stream/ref.html                    */
26      /*         and incorporated herein by reference.                         */
27      /*         As the copyright holder, John McCalpin retains the            */
28      /*         right to determine conformity with the Run Rules.             */
29      /*     3b. Results based on modified source code or on runs not in       */
30      /*         accordance with the STREAM Run Rules must be clearly          */
31      /*         labelled whenever they are published.  Examples of            */
32      /*         proper labelling include:                                     */
33      /*         "tuned STREAM benchmark results"                              */
34      /*         "based on a variant of the STREAM benchmark code"             */
35      /*         Other comparable, clear and reasonable labelling is           */
36      /*         acceptable.                                                   */
37      /*     3c. Submission of results to the STREAM benchmark web site        */
38      /*         is encouraged, but not required.                              */
39      /*  4. Use of this program or creation of derived works based on this    */
40      /*     program constitutes acceptance of these licensing restrictions.   */
41      /*  5. Absolutely no warranty is expressed or implied.                   */
42      /*-----------------------------------------------------------------------*/
43      # include <stdio.h>
44      # include <math.h>
45      # include <float.h>
46      # include <limits.h>
47      # include <sys/time.h>
```

```
48
49      /* INSTRUCTIONS:
50       *
51       *      1) Stream requires a good bit of memory to run.  Adjust the
52       *          value of 'N' (below) to give a 'timing calibration' of
53       *          at least 20 clock-ticks.  This will provide rate estimates
54       *          that should be good to about 5% precision.
55       */
56
57      #ifndef N
58      #   define N    40000000
59      #endif
60      #ifndef NTIMES
61      #   define NTIMES       10
62      #endif
63      #ifndef OFFSET
64      #   define OFFSET       0
65      #endif
66
67      /*
68       *      3) Compile the code with full optimization.  Many compilers
69       *         generate unreasonably bad code before the optimizer tightens
70       *         things up.  If the results are unreasonably good, on the
71       *         other hand, the optimizer might be too smart for me!
72       *
73       *         Try compiling with:
74       *               cc -O stream_omp.c -o stream_omp
75       *
76       *         This is known to work on Cray, SGI, IBM, and Sun machines.
77       *
78       *
79       *      4) Mail the results to mccalpin@cs.virginia.edu
80       *         Be sure to include:
81       *              a) computer hardware model number and software revision
82       *              b) the compiler flags
83       *              c) all of the output from the test case.
84       * Thanks!
85       *
86       */
87
88      # define HLINE "-------------------------------------------------------------\n"
89
90      # ifndef MIN
91      # define MIN(x,y) ((x)<(y)?(x):(y))
92      # endif
93      # ifndef MAX
94      # define MAX(x,y) ((x)>(y)?(x):(y))
95      # endif
96
97      static double   a[N+OFFSET],
98                      b[N+OFFSET],
99                      c[N+OFFSET];
100
101     static double   avgtime[4] = {0}, maxtime[4] = {0},
102                     mintime[4] = {FLT_MAX,FLT_MAX,FLT_MAX,FLT_MAX};
103
104     static char     *label[4] = {"Copy:      ", "Scale:     ",
105         "Add:       ", "Triad:     "};
106
107     static double   bytes[4] = {
108         2 * sizeof(double) * N,
109         2 * sizeof(double) * N,
110         3 * sizeof(double) * N,
111         3 * sizeof(double) * N
112         };
113
114     extern double mysecond();
115     extern void checkSTREAMresults();
116     #ifdef TUNED
117     extern void tuned_STREAM_Copy();
118     extern void tuned_STREAM_Scale(double scalar);
```

```
119     extern void tuned_STREAM_Add();
120     extern void tuned_STREAM_Triad(double scalar);
121     #endif
122     #ifdef _OPENMP
123     extern int omp_get_num_threads();
124     #endif
125     int
126     main()
127         {
//INLINE REPORT: (main()) [1] /mnt/a/u/staff/arnoldg/stream/stream.c(127,5)
//  -> INLINE: (182,22) checktick()
//    -> INLINE: (298,7) mysecond()
//    -> INLINE: (299,14) mysecond()
//    -> INLINE: (299,14) mysecond()
//  -> INLINE: (191,9) mysecond()
//  -> INLINE: (195,18) mysecond()
//  -> INLINE: (215,16) mysecond()
//  -> INLINE: (217,9) tuned_STREAM_Copy()
//  -> INLINE: (223,16) mysecond()
//  -> INLINE: (225,16) mysecond()
//  -> INLINE: (227,9) tuned_STREAM_Scale(double)
//  -> INLINE: (233,16) mysecond()
//  -> INLINE: (235,16) mysecond()
//  -> INLINE: (237,9) tuned_STREAM_Add()
//  -> INLINE: (243,16) mysecond()
//  -> INLINE: (245,16) mysecond()
//  -> INLINE: (247,9) tuned_STREAM_Triad(double)
//  -> INLINE: (253,16) mysecond()
//  -> INLINE: (281,5) checkSTREAMresults()
//
///mnt/a/u/staff/arnoldg/stream/stream.c(127,5):remark #34051: REGISTER ALLOCATION : [main] /mnt/a/u/staff
/arnoldg/stream/stream.c:127
//
//    Hardware registers
//        Reserved     :    2[ rsp rip]
//        Available    :   39[ rax rdx rcx rbx rbp rsi rdi r8-r15 mm0-mm7 zmm0-zmm15]
//        Callee-save  :    6[ rbx rbp r12-r15]
//        Assigned     :   30[ rax rdx rcx rbx rsi rdi r8-r15 zmm0-zmm15]
//
//    Routine temporaries
//        Total         :    1219
//            Global    :     168
//            Local     :    1051
//        Regenerable   :     448
//        Spilled       :      11
//
//    Routine stack
//        Variables     :     916 bytes*
//            Reads     :     124 [3.24e+02 ~ 0.0%]
//            Writes    :      41 [1.40e+01 ~ 0.0%]
//        Spills        :     128 bytes*
//            Reads     :      63 [0.00e+00 ~ 0.0%]
//            Writes    :      57 [1.00e+01 ~ 0.0%]
//
//    Notes
//
//        *Non-overlapping variables and spills may share stack space,
//         so the total stack size might be less than this.
//
//
128         int                 quantum, checktick();
129         int                 BytesPerWord;
130         register int        j, k;
131         double              scalar, t, times[4][NTIMES];
132
133         /* --- SETUP --- determine precision and check timing --- */
134
135         printf(HLINE);
136         printf("STREAM version $Revision: 5.9 $\n");
137         printf(HLINE);
138         BytesPerWord = sizeof(double);
```

```
139         printf("This system uses %d bytes per DOUBLE PRECISION word.\n",
140             BytesPerWord);
141 
142         printf(HLINE);
143     #ifdef NO_LONG_LONG
144         printf("Array size = %d, Offset = %d\n" , N, OFFSET);
145     #else
146         printf("Array size = %llu, Offset = %d\n", (unsigned long long) N, OFFSET);
147     #endif
148 
149         printf("Total memory required = %.1f MB.\n",
150             (3.0 * BytesPerWord) * ( (double) N / 1048576.0));
151         printf("Each test is run %d times, but only\n", NTIMES);
152         printf("the *best* time for each is used.\n");
153 
154     #ifdef _OPENMP
155         printf(HLINE);
156     #pragma omp parallel
//OpenMP Construct at /mnt/a/u/staff/arnoldg/stream/stream.c(156,1)
//remark #16201: OpenMP DEFINED REGION WAS PARALLELIZED
157         {
158     #pragma omp master
//OpenMP Construct at /mnt/a/u/staff/arnoldg/stream/stream.c(158,1)
//remark #16205: OpenMP multithreaded code generation for MASTER was successful
159             {
160                 k = omp_get_num_threads();
161                 printf ("Number of Threads requested = %i\n",k);
162             }
163         }
164     #endif
165 
166         printf(HLINE);
167     #pragma omp parallel
//OpenMP Construct at /mnt/a/u/staff/arnoldg/stream/stream.c(167,1)
//remark #16201: OpenMP DEFINED REGION WAS PARALLELIZED
168         {
169         printf ("Printing one line per active thread....\n");
170         }
171 
172         /* Get initial value for system clock. */
173     #pragma omp parallel for
//OpenMP Construct at /mnt/a/u/staff/arnoldg/stream/stream.c(173,1)
//remark #16200: OpenMP DEFINED LOOP WAS PARALLELIZED
//
//LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(173,1)
//<Peeled loop for vectorization>
//LOOP END
//
//LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(173,1)
//   remark #15300: LOOP WAS VECTORIZED
//LOOP END
//
//LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(173,1)
//<Remainder loop for vectorization>
//LOOP END
174         for (j=0; j<N; j++) {
175             a[j] = 1.0;
176             b[j] = 2.0;
177             c[j] = 0.0;
178             }
179 
180         printf(HLINE);
181 
182         if  ( (quantum = checktick()) >= 1)
//
//LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(297,5) inlined into /mnt/a/u/staff/arnoldg/stream/stream.
c(182,22)
//   remark #15344: loop was not vectorized: vector dependence prevents vectorization. First dependence is
shown below. Use level 5 report for details
//   remark #15346: vector dependence: assumed OUTPUT dependence between call:gettimeofday(struct timeval
*__restrict__, __timezone_ptr_t (332:13) and call:gettimeofday(struct timeval *__restrict__, __timezone_ptr_t
```

```
(332:13)
//
//   LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(299,2) inlined into /mnt/a/u/staff/arnoldg/stream
/stream.c(182,22)
//      remark #15521: loop was not vectorized: loop control variable was not identified. Explicitly compute
the iteration count before executing the loop or try using canonical loop form from OpenMP specification
//   LOOP END
//LOOP END
//
//LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(311,5) inlined into /mnt/a/u/staff/arnoldg/stream/stream.
c(182,22)
//   remark #15300: LOOP WAS VECTORIZED
//LOOP END
//
//LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(311,5) inlined into /mnt/a/u/staff/arnoldg/stream/stream.
c(182,22)
//<Remainder loop for vectorization>
//   remark #25436: completely unrolled by 3
//LOOP END
183             printf("Your clock granularity/precision appears to be "
184                 "%d microseconds.\n", quantum);
185         else {
186             printf("Your clock granularity appears to be "
187                 "less than one microsecond.\n");
188             quantum = 1;
189         }
190
191         t = mysecond();
192     #pragma omp parallel for
//OpenMP Construct at /mnt/a/u/staff/arnoldg/stream/stream.c(192,1)
//remark #16200: OpenMP DEFINED LOOP WAS PARALLELIZED
//
//LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(192,1)
//<Peeled loop for vectorization>
//LOOP END
//
//LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(192,1)
//   remark #15300: LOOP WAS VECTORIZED
//LOOP END
//
//LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(192,1)
//<Remainder loop for vectorization>
//LOOP END
193         for (j = 0; j < N; j++)
194             a[j] = 2.0E0 * a[j];
195         t = 1.0E6 * (mysecond() - t);
196
197         printf("Each test below will take on the order"
198             " of %d microseconds.\n", (int) t  );
199         printf("   (= %d clock ticks)\n", (int) (t/quantum) );
200         printf("Increase the size of the arrays if this shows that\n");
201         printf("you are not getting at least 20 clock ticks per test.\n");
202
203         printf(HLINE);
204
205         printf("WARNING -- The above is only a rough guideline.\n");
206         printf("For best results, please be sure you know the\n");
207         printf("precision of your system timer.\n");
208         printf(HLINE);
209
210         /*  --- MAIN LOOP --- repeat test cases NTIMES times --- */
211
212         scalar = 3.0;
213         for (k=0; k<NTIMES; k++)
//
//LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(213,5)
//   remark #15521: loop was not vectorized: loop control variable was not identified. Explicitly compute the
iteration count before executing the loop or try using canonical loop form from OpenMP specification
//LOOP END
214             {
215             times[0][k] = mysecond();
```

```
216     #ifdef TUNED
217             tuned_STREAM_Copy();
//
//LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(404,1) inlined into /mnt/a/u/staff/arnoldg/stream/stream.
c(217,9)
//   remark #25399: memcopy generated
//   remark #15398: loop was not vectorized: loop was transformed to memset or memcpy
//
//   LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(404,1) inlined into /mnt/a/u/staff/arnoldg/stream
/stream.c(217,9)
//      remark #15335: loop was not vectorized: vectorization possible but seems inefficient. Use vector always
directive or -vec-threshold0 to override
//      remark #25439: unrolled with remainder by 2
//   LOOP END
//
//   LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(404,1) inlined into /mnt/a/u/staff/arnoldg/stream
/stream.c(217,9)
//   <Remainder>
//   LOOP END
//LOOP END
218     #else
219     #pragma omp parallel for
220             for (j=0; j<N; j++)
221                 c[j] = a[j];
222     #endif
223             times[0][k] = mysecond() - times[0][k];
224
225             times[1][k] = mysecond();
226     #ifdef TUNED
227             tuned_STREAM_Scale(scalar);
//
//LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(412,1) inlined into /mnt/a/u/staff/arnoldg/stream/stream.
c(227,9)
//<Peeled loop for vectorization>
//LOOP END
//
//LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(412,1) inlined into /mnt/a/u/staff/arnoldg/stream/stream.
c(227,9)
//   remark #15300: LOOP WAS VECTORIZED
//LOOP END
//
//LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(412,1) inlined into /mnt/a/u/staff/arnoldg/stream/stream.
c(227,9)
//<Remainder loop for vectorization>
//LOOP END
228     #else
229     #pragma omp parallel for
230             for (j=0; j<N; j++)
231                 b[j] = scalar*c[j];
232     #endif
233             times[1][k] = mysecond() - times[1][k];
234
235             times[2][k] = mysecond();
236     #ifdef TUNED
237             tuned_STREAM_Add();
//
//LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(420,1) inlined into /mnt/a/u/staff/arnoldg/stream/stream.
c(237,9)
//<Peeled loop for vectorization>
//LOOP END
//
//LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(420,1) inlined into /mnt/a/u/staff/arnoldg/stream/stream.
c(237,9)
//   remark #15300: LOOP WAS VECTORIZED
//LOOP END
//
//LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(420,1) inlined into /mnt/a/u/staff/arnoldg/stream/stream.
c(237,9)
//<Remainder loop for vectorization>
//LOOP END
238     #else
```

```
239     #pragma omp parallel for
240             for (j=0; j<N; j++)
241                 c[j] = a[j]+b[j];
242     #endif
243             times[2][k] = mysecond() - times[2][k];
244
245             times[3][k] = mysecond();
246     #ifdef TUNED
247             tuned_STREAM_Triad(scalar);
//
//LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(428,1) inlined into /mnt/a/u/staff/arnoldg/stream/stream.
c(247,9)
//<Peeled loop for vectorization>
//LOOP END
//
//LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(428,1) inlined into /mnt/a/u/staff/arnoldg/stream/stream.
c(247,9)
//   remark #15300: LOOP WAS VECTORIZED
//LOOP END
//
//LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(428,1) inlined into /mnt/a/u/staff/arnoldg/stream/stream.
c(247,9)
//<Remainder loop for vectorization>
//LOOP END
248     #else
249     #pragma omp parallel for
250             for (j=0; j<N; j++)
251                 a[j] = b[j]+scalar*c[j];
252     #endif
253             times[3][k] = mysecond() - times[3][k];
254             }
255
256         /*  --- SUMMARY --- */
257
258         for (k=1; k<NTIMES; k++) /* note -- skip first iteration */
//
//LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(258,5)
//   remark #25461: Imperfect Loop Unroll-Jammed by 4   (pre-vector)
//   remark #25045: Fused Loops: ( 258 258 )
//
//   remark #25084: Preprocess Loopnests: Moving Out Store    [ /mnt/a/u/staff/arnoldg/stream/stream.c(258,25) ]
//   remark #15335: loop was not vectorized: vectorization possible but seems inefficient. Use vector always
directive or -vec-threshold0 to override
//   remark #25436: completely unrolled by 9
//LOOP END
//
//LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(258,5)
//<Distributed chunk2>
//   remark #25046: Loop lost in Fusion
//LOOP END
259             {
260             for (j=0; j<4; j++)
261                 {
262                 avgtime[j] = avgtime[j] + times[j][k];
263                 mintime[j] = MIN(mintime[j], times[j][k]);
264                 maxtime[j] = MAX(maxtime[j], times[j][k]);
265                 }
266             }
267
268         printf("Function      Rate (MB/s)   Avg time     Min time     Max time\n");
269         for (j=0; j<4; j++) {
//
//LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(269,5)
//   remark #15344: loop was not vectorized: vector dependence prevents vectorization. First dependence is
shown below. Use level 5 report for details
//   remark #25436: completely unrolled by 4
//LOOP END
270             avgtime[j] = avgtime[j]/(double)(NTIMES-1);
271
272             printf("%s%11.4f  %11.4f  %11.4f  %11.4f\n", label[j],
273                    1.0E-06 * bytes[j]/mintime[j],
```

```
274                     avgtime[j],
275                     mintime[j],
276                     maxtime[j]);
277         }
278         printf(HLINE);
279 
280         /* --- Check Results --- */
281         checkSTREAMresults();
//
//LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(351,2) inlined into /mnt/a/u/staff/arnoldg/stream/stream.
c(281,5)
//   remark #15344: loop was not vectorized: vector dependence prevents vectorization. First dependence is
shown below. Use level 5 report for details
//   remark #15346: vector dependence: assumed ANTI dependence between aj (354:13) and aj (356:13)
//   remark #25436: completely unrolled by 10
//LOOP END
//
//LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(365,2) inlined into /mnt/a/u/staff/arnoldg/stream/stream.
c(281,5)
//   remark #15300: LOOP WAS VECTORIZED
//LOOP END
282         printf(HLINE);
283 
284         return 0;
285     }
286 
287     # define        M       20
288 
289     int
290     checktick()
291         {
//INLINE REPORT: (checktick()) [2] /mnt/a/u/staff/arnoldg/stream/stream.c(291,5)
//  -> INLINE: (298,7) mysecond()
//  -> INLINE: (299,14) mysecond()
//  -> INLINE: (299,14) mysecond()
//
///mnt/a/u/staff/arnoldg/stream/stream.c(291,5):remark #34051: REGISTER ALLOCATION : [checktick] /mnt/a/u/staff
/arnoldg/stream/stream.c:291
//
//    Hardware registers
//        Reserved     :    2[ rsp rip]
//        Available    :   39[ rax rdx rcx rbx rbp rsi rdi r8-r15 mm0-mm7 zmm0-zmm15]
//        Callee-save  :    6[ rbx rbp r12-r15]
//        Assigned     :   22[ rax rdx rcx rbp rsi rdi zmm0-zmm15]
//
//    Routine temporaries
//        Total         :     144
//            Global    :      16
//            Local     :     128
//        Regenerable   :      18
//        Spilled       :       2
//
//    Routine stack
//        Variables     :     232 bytes*
//            Reads     :      26 [3.00e+02 ~ 12.0%]
//            Writes    :       1 [2.00e+01 ~ 0.8%]
//        Spills        :       8 bytes*
//            Reads     :       2 [1.20e+02 ~ 4.8%]
//            Writes    :       1 [2.00e+01 ~ 0.8%]
//
//    Notes
//
//        *Non-overlapping variables and spills may share stack space,
//         so the total stack size might be less than this.
//
//
292         int         i, minDelta, Delta;
293         double      t1, t2, timesfound[M];
294 
295     /*  Collect a sequence of M unique time values from the system. */
296 
```

```
297         for (i = 0; i < M; i++) {
//
//LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(297,5)
//   remark #15344: loop was not vectorized: vector dependence prevents vectorization. First dependence is
shown below. Use level 5 report for details
//   remark #15346: vector dependence: assumed OUTPUT dependence between call:gettimeofday(struct timeval
*__restrict__, __timezone_ptr_t (332:13) and call:gettimeofday(struct timeval *__restrict__, __timezone_ptr_t
(332:13)
//
//   LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(299,2)
//      remark #15521: loop was not vectorized: loop control variable was not identified. Explicitly compute
the iteration count before executing the loop or try using canonical loop form from OpenMP specification
//   LOOP END
//LOOP END
298             t1 = mysecond();
299             while( ((t2=mysecond()) - t1) < 1.0E-6 )
300                 ;
301             timesfound[i] = t1 = t2;
302             }
303
304     /*
305      * Determine the minimum difference between these M values.
306      * This result will be our estimate (in microseconds) for the
307      * clock granularity.
308      */
309
310         minDelta = 1000000;
311         for (i = 1; i < M; i++) {
//
//LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(311,5)
//   remark #15300: LOOP WAS VECTORIZED
//LOOP END
//
//LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(311,5)
//<Remainder loop for vectorization>
//   remark #25436: completely unrolled by 3
//LOOP END
312             Delta = (int)( 1.0E6 * (timesfound[i]-timesfound[i-1]));
///mnt/a/u/staff/arnoldg/stream/stream.c(312,26):remark #34055: adjacent dense (unit-strided stencil) loads are
not optimized. Details: stride { 8 }, step { 8 }, types { F64-V128, F64-V128 }, number of elements { 2 },
select mask { 0x000000003 }.
///mnt/a/u/staff/arnoldg/stream/stream.c(312,26):remark #34055: adjacent dense (unit-strided stencil) loads are
not optimized. Details: stride { 8 }, step { 8 }, types { F64-V128, F64-V128 }, number of elements { 2 },
select mask { 0x000000003 }.
///mnt/a/u/staff/arnoldg/stream/stream.c(312,26):remark #34055: adjacent dense (unit-strided stencil) loads are
not optimized. Details: stride { 8 }, step { 8 }, types { F64-V128, F64-V128 }, number of elements { 2 },
select mask { 0x000000003 }.
///mnt/a/u/staff/arnoldg/stream/stream.c(312,26):remark #34055: adjacent dense (unit-strided stencil) loads are
not optimized. Details: stride { 8 }, step { 8 }, types { F64-V128, F64-V128 }, number of elements { 2 },
select mask { 0x000000003 }.
///mnt/a/u/staff/arnoldg/stream/stream.c(312,26):remark #34055: adjacent dense (unit-strided stencil) loads are
not optimized. Details: stride { 8 }, step { 8 }, types { F64-V128, F64-V128 }, number of elements { 2 },
select mask { 0x000000003 }.
///mnt/a/u/staff/arnoldg/stream/stream.c(312,26):remark #34055: adjacent dense (unit-strided stencil) loads are
not optimized. Details: stride { 8 }, step { 8 }, types { F64-V128, F64-V128 }, number of elements { 2 },
select mask { 0x000000003 }.
///mnt/a/u/staff/arnoldg/stream/stream.c(312,26):remark #34055: adjacent dense (unit-strided stencil) loads are
not optimized. Details: stride { 8 }, step { 8 }, types { F64-V128, F64-V128 }, number of elements { 2 },
select mask { 0x000000003 }.
///mnt/a/u/staff/arnoldg/stream/stream.c(312,26):remark #34055: adjacent dense (unit-strided stencil) loads are
not optimized. Details: stride { 8 }, step { 8 }, types { F64-V128, F64-V128 }, number of elements { 2 },
select mask { 0x000000003 }.
///mnt/a/u/staff/arnoldg/stream/stream.c(312,26):remark #34055: adjacent dense (unit-strided stencil) loads are
not optimized. Details: stride { 8 }, step { 8 }, types { F64-V128, F64-V128 }, number of elements { 2 },
select mask { 0x000000003 }.
///mnt/a/u/staff/arnoldg/stream/stream.c(312,26):remark #34055: adjacent dense (unit-strided stencil) loads are
not optimized. Details: stride { 8 }, step { 8 }, types { F64-V128, F64-V128 }, number of elements { 2 },
select mask { 0x000000003 }.
///mnt/a/u/staff/arnoldg/stream/stream.c(312,26):remark #34055: adjacent dense (unit-strided stencil) loads are
not optimized. Details: stride { 8 }, step { 8 }, types { F64-V128, F64-V128 }, number of elements { 2 },
select mask { 0x000000003 }.
```

```
///mnt/a/u/staff/arnoldg/stream/stream.c(312,26):remark #34055: adjacent dense (unit-strided stencil) loads are not optimized. Details: stride { 8 }, step { 8 }, types { F64-V128, F64-V128 }, number of elements { 2 }, select mask { 0x000000003 }.
///mnt/a/u/staff/arnoldg/stream/stream.c(312,26):remark #34055: adjacent dense (unit-strided stencil) loads are not optimized. Details: stride { 8 }, step { 8 }, types { F64-V128, F64-V128 }, number of elements { 2 }, select mask { 0x000000003 }.
///mnt/a/u/staff/arnoldg/stream/stream.c(312,26):remark #34055: adjacent dense (unit-strided stencil) loads are not optimized. Details: stride { 8 }, step { 8 }, types { F64-V128, F64-V128 }, number of elements { 2 }, select mask { 0x000000003 }.
///mnt/a/u/staff/arnoldg/stream/stream.c(312,26):remark #34055: adjacent dense (unit-strided stencil) loads are not optimized. Details: stride { 8 }, step { 8 }, types { F64-V128, F64-V128 }, number of elements { 2 }, select mask { 0x000000003 }.
///mnt/a/u/staff/arnoldg/stream/stream.c(312,26):remark #34055: adjacent dense (unit-strided stencil) loads are not optimized. Details: stride { 8 }, step { 8 }, types { F64-V128, F64-V128 }, number of elements { 2 }, select mask { 0x000000003 }.
313             minDelta = MIN(minDelta, MAX(Delta,0));
314             }
315
316        return(minDelta);
317         }
318
319
320
321     /* A gettimeofday routine to give access to the wall
322        clock timer on most UNIX-like systems.  */
323
324     #include <sys/time.h>
325
326     double mysecond()
327     {
//INLINE REPORT: (mysecond()) [3] /mnt/a/u/staff/arnoldg/stream/stream.c(327,1)
//
///mnt/a/u/staff/arnoldg/stream/stream.c(327,1):remark #34051: REGISTER ALLOCATION : [mysecond] /mnt/a/u/staff/arnoldg/stream/stream.c:327
//
//    Hardware registers
//        Reserved     :    2[ rsp rip]
//        Available    :   39[ rax rdx rcx rbx rbp rsi rdi r8-r15 mm0-mm7 zmm0-zmm15]
//        Callee-save  :    6[ rbx rbp r12-r15]
//        Assigned     :    4[ rsi rdi zmm0-zmm1]
//
//    Routine temporaries
//        Total         :      15
//            Global    :       6
//            Local     :       9
//        Regenerable   :       4
//        Spilled       :       0
//
//    Routine stack
//        Variables     :      24 bytes*
//            Reads     :       2 [2.00e+00 ~ 9.1%]
//            Writes    :       0 [0.00e+00 ~ 0.0%]
//        Spills        :       0 bytes*
//            Reads     :       0 [0.00e+00 ~ 0.0%]
//            Writes    :       0 [0.00e+00 ~ 0.0%]
//
//    Notes
//
//        *Non-overlapping variables and spills may share stack space,
//         so the total stack size might be less than this.
//
//
328             struct timeval tp;
329             struct timezone tzp;
330             int i;
331
332             i = gettimeofday(&tp,&tzp);
333             return ( (double) tp.tv_sec + (double) tp.tv_usec * 1.e-6 );
334     }
335
336     void checkSTREAMresults ()
```

```
337     {
//INLINE REPORT: (checkSTREAMresults()) [4] /mnt/a/u/staff/arnoldg/stream/stream.c(337,1)
//
///mnt/a/u/staff/arnoldg/stream/stream.c(337,1):remark #34051: REGISTER ALLOCATION : [checkSTREAMresults] /mnt/a
/u/staff/arnoldg/stream/stream.c:337
//
//    Hardware registers
//        Reserved     :    2[ rsp rip]
//        Available    :   39[ rax rdx rcx rbx rbp rsi rdi r8-r15 mm0-mm7 zmm0-zmm15]
//        Callee-save  :    6[ rbx rbp r12-r15]
//        Assigned     :    9[ rax rdi zmm0-zmm6]
//
//    Routine temporaries
//        Total         :      73
//            Global    :      17
//            Local     :      56
//        Regenerable   :      33
//        Spilled       :       3
//
//    Routine stack
//        Variables     :       0 bytes*
//            Reads     :       0 [0.00e+00 ~ 0.0%]
//            Writes    :       0 [0.00e+00 ~ 0.0%]
//        Spills        :      24 bytes*
//            Reads     :       4 [0.00e+00 ~ 0.0%]
//            Writes    :       3 [0.00e+00 ~ 0.0%]
//
//    Notes
//
//        *Non-overlapping variables and spills may share stack space,
//         so the total stack size might be less than this.
//
//
338             double aj,bj,cj,scalar;
339             double asum,bsum,csum;
340             double epsilon;
341             int     j,k;
342
343         /* reproduce initialization */
344             aj = 1.0;
345             bj = 2.0;
346             cj = 0.0;
347         /* a[] is modified during timing check */
348             aj = 2.0E0 * aj;
349         /* now execute timing loop */
350             scalar = 3.0;
351             for (k=0; k<NTIMES; k++)
//
//LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(351,2)
//   remark #15344: loop was not vectorized: vector dependence prevents vectorization. First dependence is
shown below. Use level 5 report for details
//   remark #15346: vector dependence: assumed ANTI dependence between aj (354:13) and aj (356:13)
//   remark #25436: completely unrolled by 10
//LOOP END
352             {
353                 cj = aj;
354                 bj = scalar*cj;
355                 cj = aj+bj;
356                 aj = bj+scalar*cj;
357             }
358             aj = aj * (double) (N);
359             bj = bj * (double) (N);
360             cj = cj * (double) (N);
361
362             asum = 0.0;
363             bsum = 0.0;
364             csum = 0.0;
365             for (j=0; j<N; j++) {
//
//LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(365,2)
//   remark #15300: LOOP WAS VECTORIZED
```

```
//LOOP END
366                     asum += a[j];
367                     bsum += b[j];
368                     csum += c[j];
369             }
370     #ifdef VERBOSE
371             printf ("Results Comparison: \n");
372             printf ("        Expected  : %f %f %f \n",aj,bj,cj);
373             printf ("        Observed  : %f %f %f \n",asum,bsum,csum);
374     #endif
375
376     #ifndef abs
377     #define abs(a) ((a) >= 0 ? (a) : -(a))
378     #endif
379             epsilon = 1.e-8;
380
381             if (abs(aj-asum)/asum > epsilon) {
382                     printf ("Failed Validation on array a[]\n");
383                     printf ("        Expected  : %f \n",aj);
384                     printf ("        Observed  : %f \n",asum);
385             }
386             else if (abs(bj-bsum)/bsum > epsilon) {
387                     printf ("Failed Validation on array b[]\n");
388                     printf ("        Expected  : %f \n",bj);
389                     printf ("        Observed  : %f \n",bsum);
390             }
391             else if (abs(cj-csum)/csum > epsilon) {
392                     printf ("Failed Validation on array c[]\n");
393                     printf ("        Expected  : %f \n",cj);
394                     printf ("        Observed  : %f \n",csum);
395             }
396             else {
397                     printf ("Solution Validates\n");
398             }
399     }
400
401     void tuned_STREAM_Copy()
402     {
//INLINE REPORT: (tuned_STREAM_Copy()) [5] /mnt/a/u/staff/arnoldg/stream/stream.c(402,1)
//
///mnt/a/u/staff/arnoldg/stream/stream.c(402,1):remark #34051: REGISTER ALLOCATION : [tuned_STREAM_Copy] /mnt/a
/u/staff/arnoldg/stream/stream.c:402
//
//    Hardware registers
//        Reserved     :    2[ rsp rip]
//        Available    :   39[ rax rdx rcx rbx rbp rsi rdi r8-r15 mm0-mm7 zmm0-zmm15]
//        Callee-save  :    6[ rbx rbp r12-r15]
//        Assigned     :   10[ rax rdx rcx rbx rbp rsi rdi r8-r10]
//
//    Routine temporaries
//        Total         :      94
//            Global    :      18
//            Local     :      76
//        Regenerable   :      32
//        Spilled       :       0
//
//    Routine stack
//        Variables     :      20 bytes*
//            Reads     :       4 [0.00e+00 ~ 0.0%]
//            Writes    :       5 [5.00e+00 ~ 0.0%]
//        Spills        :      48 bytes*
//            Reads     :      12 [0.00e+00 ~ 0.0%]
//            Writes    :      12 [1.20e+01 ~ 0.0%]
//
//    Notes
//
//        *Non-overlapping variables and spills may share stack space,
//         so the total stack size might be less than this.
//
//
403             int j;
```

```
404     #pragma omp parallel for
//OpenMP Construct at /mnt/a/u/staff/arnoldg/stream/stream.c(404,1) inlined into /mnt/a/u/staff/arnoldg/stream
/stream.c(217,9)
//remark #16200: OpenMP DEFINED LOOP WAS PARALLELIZED
///mnt/a/u/staff/arnoldg/stream/stream.c(404,1):remark #34026: call to memcpy implemented as a call to
optimized library version
//OpenMP Construct at /mnt/a/u/staff/arnoldg/stream/stream.c(404,1)
//remark #16200: OpenMP DEFINED LOOP WAS PARALLELIZED
//
//LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(404,1)
//   remark #25399: memcopy generated
//   remark #15398: loop was not vectorized: loop was transformed to memset or memcpy
//
//   LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(404,1)
//      remark #15335: loop was not vectorized: vectorization possible but seems inefficient. Use vector always
directive or -vec-threshold0 to override
//      remark #25439: unrolled with remainder by 2
//   LOOP END
//
//   LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(404,1)
//   <Remainder>
//   LOOP END
//LOOP END
///mnt/a/u/staff/arnoldg/stream/stream.c(404,1):remark #34026: call to memcpy implemented as a call to
optimized library version
405             for (j=0; j<N; j++)
406                 c[j] = a[j];
407     }
408
409     void tuned_STREAM_Scale(double scalar)
410     {
//INLINE REPORT: (tuned_STREAM_Scale(double)) [6] /mnt/a/u/staff/arnoldg/stream/stream.c(410,1)
//
///mnt/a/u/staff/arnoldg/stream/stream.c(410,1):remark #34051: REGISTER ALLOCATION : [tuned_STREAM_Scale] /mnt/a
/u/staff/arnoldg/stream/stream.c:410
//
//    Hardware registers
//        Reserved     :    2[ rsp rip]
//        Available    :   39[ rax rdx rcx rbx rbp rsi rdi r8-r15 mm0-mm7 zmm0-zmm15]
//        Callee-save  :    6[ rbx rbp r12-r15]
//        Assigned     :   14[ rax rdx rcx rbx rbp rsi rdi r8-r11 zmm0-zmm2]
//
//    Routine temporaries
//        Total         :     106
//            Global    :      19
//            Local     :      87
//        Regenerable   :      37
//        Spilled       :       1
//
//    Routine stack
//        Variables     :      28 bytes*
//            Reads     :       4 [0.00e+00 ~ 0.0%]
//            Writes    :       6 [6.00e+00 ~ 0.0%]
//        Spills        :      56 bytes*
//            Reads     :      13 [1.00e+00 ~ 0.0%]
//            Writes    :      13 [1.30e+01 ~ 0.0%]
//
//    Notes
//
//        *Non-overlapping variables and spills may share stack space,
//         so the total stack size might be less than this.
//
//
411             int j;
412     #pragma omp parallel for
//OpenMP Construct at /mnt/a/u/staff/arnoldg/stream/stream.c(412,1) inlined into /mnt/a/u/staff/arnoldg/stream
/stream.c(227,9)
//remark #16200: OpenMP DEFINED LOOP WAS PARALLELIZED
//OpenMP Construct at /mnt/a/u/staff/arnoldg/stream/stream.c(412,1)
//remark #16200: OpenMP DEFINED LOOP WAS PARALLELIZED
//
```

```
//LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(412,1)
//<Peeled loop for vectorization>
//LOOP END
//
//LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(412,1)
//   remark #15300: LOOP WAS VECTORIZED
//LOOP END
//
//LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(412,1)
//<Remainder loop for vectorization>
//LOOP END
413             for (j=0; j<N; j++)
414                 b[j] = scalar*c[j];
415     }
416
417     void tuned_STREAM_Add()
418     {
//INLINE REPORT: (tuned_STREAM_Add()) [7] /mnt/a/u/staff/arnoldg/stream/stream.c(418,1)
//
///mnt/a/u/staff/arnoldg/stream/stream.c(418,1):remark #34051: REGISTER ALLOCATION : [tuned_STREAM_Add] /mnt/a/u
/staff/arnoldg/stream/stream.c:418
//
//    Hardware registers
//        Reserved     :    2[ rsp rip]
//        Available    :   39[ rax rdx rcx rbx rbp rsi rdi r8-r15 mm0-mm7 zmm0-zmm15]
//        Callee-save  :    6[ rbx rbp r12-r15]
//        Assigned     :   12[ rax rdx rcx rbx rbp rsi rdi r8-r11 zmm0]
//
//    Routine temporaries
//        Total         :      94
//            Global    :      17
//            Local     :      77
//        Regenerable   :      33
//        Spilled       :       0
//
//    Routine stack
//        Variables     :      20 bytes*
//            Reads     :       4 [0.00e+00 ~ 0.0%]
//            Writes    :       5 [5.00e+00 ~ 0.0%]
//        Spills        :      48 bytes*
//            Reads     :      12 [0.00e+00 ~ 0.0%]
//            Writes    :      12 [1.20e+01 ~ 0.0%]
//
//    Notes
//
//        *Non-overlapping variables and spills may share stack space,
//         so the total stack size might be less than this.
//
//
419             int j;
420     #pragma omp parallel for
//OpenMP Construct at /mnt/a/u/staff/arnoldg/stream/stream.c(420,1) inlined into /mnt/a/u/staff/arnoldg/stream
/stream.c(237,9)
//remark #16200: OpenMP DEFINED LOOP WAS PARALLELIZED
//OpenMP Construct at /mnt/a/u/staff/arnoldg/stream/stream.c(420,1)
//remark #16200: OpenMP DEFINED LOOP WAS PARALLELIZED
//
//LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(420,1)
//<Peeled loop for vectorization>
//LOOP END
//
//LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(420,1)
//   remark #15300: LOOP WAS VECTORIZED
//LOOP END
//
//LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(420,1)
//<Remainder loop for vectorization>
//LOOP END
421             for (j=0; j<N; j++)
422                 c[j] = a[j]+b[j];
423     }
```

```
424
425     void tuned_STREAM_Triad(double scalar)
426     {
//INLINE REPORT: (tuned_STREAM_Triad(double)) [8] /mnt/a/u/staff/arnoldg/stream/stream.c(426,1)
//
///mnt/a/u/staff/arnoldg/stream/stream.c(426,1):remark #34051: REGISTER ALLOCATION : [tuned_STREAM_Triad] /mnt/a
/u/staff/arnoldg/stream/stream.c:426
//
//    Hardware registers
//        Reserved     :    2[ rsp rip]
//        Available    :   39[ rax rdx rcx rbx rbp rsi rdi r8-r15 mm0-mm7 zmm0-zmm15]
//        Callee-save  :    6[ rbx rbp r12-r15]
//        Assigned     :   14[ rax rdx rcx rbx rbp rsi rdi r8-r11 zmm0-zmm2]
//
//    Routine temporaries
//        Total         :     108
//            Global    :      19
//            Local     :      89
//        Regenerable   :      37
//        Spilled       :       1
//
//    Routine stack
//        Variables     :      28 bytes*
//            Reads     :       4 [0.00e+00 ~ 0.0%]
//            Writes    :       6 [6.00e+00 ~ 0.0%]
//        Spills        :      56 bytes*
//            Reads     :      13 [1.00e+00 ~ 0.0%]
//            Writes    :      13 [1.30e+01 ~ 0.0%]
//
//    Notes
//
//        *Non-overlapping variables and spills may share stack space,
//         so the total stack size might be less than this.
//
//
427             int j;
428     #pragma omp parallel for
//OpenMP Construct at /mnt/a/u/staff/arnoldg/stream/stream.c(428,1) inlined into /mnt/a/u/staff/arnoldg/stream
/stream.c(247,9)
//remark #16200: OpenMP DEFINED LOOP WAS PARALLELIZED
//OpenMP Construct at /mnt/a/u/staff/arnoldg/stream/stream.c(428,1)
//remark #16200: OpenMP DEFINED LOOP WAS PARALLELIZED
//
//LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(428,1)
//<Peeled loop for vectorization>
//LOOP END
//
//LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(428,1)
//   remark #15300: LOOP WAS VECTORIZED
//LOOP END
//
//LOOP BEGIN at /mnt/a/u/staff/arnoldg/stream/stream.c(428,1)
//<Remainder loop for vectorization>
//LOOP END
429             for (j=0; j<N; j++)
430                 a[j] = b[j]+scalar*c[j];
431     }
432
arnoldg@h2ologin4:~/stream>
```

GNU (gcc)

```
arnoldg@h2ologin4:~/stream> gcc -c -fopenmp -O3 -fopt-info -DTUNED stream.c
stream.c:194:18: note: loop vectorized
stream.c:194:18: note: loop peeled for vectorization to enhance alignment
stream.c:192:9: note: loop turned into non-loop; it never loops
stream.c:192:9: note: loop turned into non-loop; it never loops.
stream.c:192:9: note: loop with 3 iterations completely unrolled
stream.c:175:7: note: Loop 1 distributed: split to 1 loops and 1 library calls.
stream.c:175:7: note: loop vectorized
stream.c:175:7: note: loop peeled for vectorization to enhance alignment
stream.c:173:9: note: loop turned into non-loop; it never loops
stream.c:173:9: note: loop turned into non-loop; it never loops.
stream.c:173:9: note: loop with 8 iterations completely unrolled
stream.c:406:21: note: Loop 1 distributed: split to 0 loops and 1 library calls.
stream.c:414:21: note: loop vectorized
stream.c:414:21: note: loop peeled for vectorization to enhance alignment
stream.c:412:9: note: loop turned into non-loop; it never loops
stream.c:412:9: note: loop turned into non-loop; it never loops.
stream.c:412:9: note: loop with 4 iterations completely unrolled
stream.c:422:14: note: loop vectorized
stream.c:422:14: note: loop peeled for vectorization to enhance alignment
stream.c:420:9: note: loop turned into non-loop; it never loops
stream.c:420:9: note: loop turned into non-loop; it never loops.
stream.c:420:9: note: loop with 3 iterations completely unrolled
stream.c:430:14: note: loop vectorized
stream.c:430:14: note: loop peeled for vectorization to enhance alignment
stream.c:428:9: note: loop turned into non-loop; it never loops
stream.c:428:9: note: loop turned into non-loop; it never loops.
stream.c:428:9: note: loop with 3 iterations completely unrolled
stream.c:311:5: note: loop vectorized
stream.c:311:5: note: loop turned into non-loop; it never loops.
stream.c:311:5: note: loop with 3 iterations completely unrolled
stream.c:290:1: note: loop turned into non-loop; it never loops
stream.c:290:1: note: loop turned into non-loop; it never loops.
stream.c:290:1: note: loop with 4 iterations completely unrolled
stream.c:351:2: note: loop turned into non-loop; it never loops.
stream.c:351:2: note: loop with 10 iterations completely unrolled
stream.c:260:2: note: loop turned into non-loop; it never loops.
stream.c:260:2: note: loop with 5 iterations completely unrolled
stream.c:258:5: note: loop turned into non-loop; it never loops.
stream.c:258:5: note: loop with 9 iterations completely unrolled
arnoldg@h2ologin4:~/stream>
```

To confirm vectorization, compile to assembly code (gcc -S or similar) and look for vector instructions. This may change with optimization levels.

counting vector instructions

```
arnoldg@h2ologin4:~/stream> gcc -c -fopenmp -O1 -fopt-info -DTUNED -S stream.c
arnoldg@h2ologin4:~/stream> grep xmm stream.s | wc -l
138
arnoldg@h2ologin4:~/stream> gcc -c -fopenmp -O3 -fopt-info -DTUNED -S stream.c
stream.c:194:18: note: loop vectorized
stream.c:194:18: note: loop peeled for vectorization to enhance alignment
stream.c:192:9: note: loop turned into non-loop; it never loops
stream.c:192:9: note: loop turned into non-loop; it never loops.
stream.c:192:9: note: loop with 3 iterations completely unrolled
stream.c:175:7: note: Loop 1 distributed: split to 1 loops and 1 library calls.
stream.c:175:7: note: loop vectorized
stream.c:175:7: note: loop peeled for vectorization to enhance alignment
stream.c:173:9: note: loop turned into non-loop; it never loops
stream.c:173:9: note: loop turned into non-loop; it never loops.
stream.c:173:9: note: loop with 8 iterations completely unrolled
stream.c:406:21: note: Loop 1 distributed: split to 0 loops and 1 library calls.
stream.c:414:21: note: loop vectorized
stream.c:414:21: note: loop peeled for vectorization to enhance alignment
stream.c:412:9: note: loop turned into non-loop; it never loops
stream.c:412:9: note: loop turned into non-loop; it never loops.
stream.c:412:9: note: loop with 4 iterations completely unrolled
stream.c:422:14: note: loop vectorized
stream.c:422:14: note: loop peeled for vectorization to enhance alignment
stream.c:420:9: note: loop turned into non-loop; it never loops
stream.c:420:9: note: loop turned into non-loop; it never loops.
stream.c:420:9: note: loop with 3 iterations completely unrolled
stream.c:430:14: note: loop vectorized
stream.c:430:14: note: loop peeled for vectorization to enhance alignment
stream.c:428:9: note: loop turned into non-loop; it never loops
stream.c:428:9: note: loop turned into non-loop; it never loops.
stream.c:428:9: note: loop with 3 iterations completely unrolled
stream.c:311:5: note: loop vectorized
stream.c:311:5: note: loop turned into non-loop; it never loops.
stream.c:311:5: note: loop with 3 iterations completely unrolled
stream.c:290:1: note: loop turned into non-loop; it never loops
stream.c:290:1: note: loop turned into non-loop; it never loops.
stream.c:290:1: note: loop with 4 iterations completely unrolled
stream.c:351:2: note: loop turned into non-loop; it never loops.
stream.c:351:2: note: loop with 10 iterations completely unrolled
stream.c:260:2: note: loop turned into non-loop; it never loops.
stream.c:260:2: note: loop with 5 iterations completely unrolled
stream.c:258:5: note: loop turned into non-loop; it never loops.
stream.c:258:5: note: loop with 9 iterations completely unrolled
arnoldg@h2ologin4:~/stream> grep xmm stream.s | wc -l
524
arnoldg@h2ologin4:~/stream>
```