ECE459 Assignment 3

Michael Liu (20306823)
Eddie Hao (20293463)

Baseline Performance

Profiler: For this assignment, perf was used to profile the code and verify its runtime. In order to determine the time spent in each portion of the code, the command *perf annotate* was used to display the results obtained through *perf record.*

Runtime: The runtime was obtained using *perf stat ./test_harness*. The results indicate that the total runtime is 25.550951754 seconds with most CPU cycles stalled.

```
Performance counter stats for './test_harness':

      25518.747654 task-clock                #    0.999 CPUs utilized
             2,587 context-switches          #    0.101 K/sec
                 0 cpu-migrations            #    0.000 K/sec
             4,831 page-faults               #    0.189 K/sec
    96,619,310,848 cycles                    #    3.786 GHz                      [83.34%]
    54,264,502,367 stalled-cycles-frontend   #   56.16% frontend cycles idle     [83.32%]
    22,569,327,322 stalled-cycles-backend    #   23.36% backend  cycles idle     [66.68%]
   124,626,870,289 instructions              #    1.29  insns per cycle
                                             #    0.44  stalled cycles per insn [83.35%]
    16,749,861,193 branches                  #  656.375 M/sec                    [83.33%]
       399,424,126 branch-misses             #    2.38% of all branches          [83.33%]

      25.550951754 seconds time elapsed
```

Profiler Output: The full output of the file is quite large and has been included in the appendix section.

Key Sections: Some key sections have been identified that consume most of the runtime during the test_harness run. Due to Amdahl's Law, only the sections that consume more than 5% of the runtime will be discussed since they are the sections which will yield the most improvement overall. All of the following sections are from model.cpp.

Section 1:

```
       |                    // Calculate u, v
       |                    u = QVector2D::dotProduct(XP, QP) /  QP.lengthSquared();
  0.42 |       divsd  %xmm0,%xmm1
 11.91 |       movsd  %xmm1,-0x78(%rbp)
```

This section consumes ~12% of the runtime and is by a large margin the largest consumer of time in the entire program. This large time consumption is caused by two major factors. First, division in C runs in $O(n^2)$ with what is most likely at least one large number (QP.lengthSquared()). Second, the function call to QVector2D::dotProduct(XP, QP) multiplies two vectors which can be an expensive operation if the vectors are large. Finally, the QP.lengthSquared() method calculates the dot product of the vector with itself, which is also an expensive operation. This section is found at line 126 of the original model.cpp file.

Section 2:

```
       |                    // get interpolating lines from reference line
       |                    QPoint P2 = listAux[h]->at(k).first;
  7.73 |       mov    (%rax,%rcx,1),%rdx
       |       mov    (%rdx),%rcx
```

This section consumes ~8% of the runtime. The code is accessing a value stored in a specific location in a vector and assigning that value to the variable P2. Although array and vector access can be cheap operations, this action can become quite expensive if performed many times. This section is found at

line 117 of the original model.cpp file.

Section 3:

```
        |      inline const QVector2D operator/(const QVector2D &vector, qreal divisor)
        |      {
        |          return QVector2D(vector.xp / divisor, vector.yp / divisor, 1);
  0.40  |        divsd  %xmm0,%xmm1
  6.87  |        unpckl %xmm1,%xmm1
  0.42  |        cvtpd2 %xmm1,%xmm1
        |          return v1.xp != v2.xp || v1.yp != v2.yp;
        |      }
```

Unpack is an expensive operation that needs to be done whenever a vector is divided by a real number.

Section 4:

```
        |      inline const QVector2D operator/(const QVector2D &vector, qreal divisor)
        |      {
        |          return QVector2D(vector.xp / divisor, vector.yp / divisor, 1);
  0.47  |        unpckl %xmm2,%xmm2
        |        cvtps2 %xmm2,%xmm2
        |        divsd  %xmm0,%xmm2
  5.91  |        unpckl %xmm2,%xmm2
  0.36  |        cvtpd2 %xmm2,%xmm0
        |          return QVector2D(v1.xp - v2.xp, v1.yp - v2.yp, 1);
        |      }
```

Unpack is an expensive operation that needs to be done whenever a vector is divided by a real number.

Improvements

Improvement 1: Structural Improvements

– moved wsum/sum calculations into innermost loop
– pre-computed length() and lengthSquared() values

Code diff:

```
--- model_orig.cpp 2013-03-23 20:19:58.221863646 -0400
+++ model_intermediate.cpp 2013-03-24 01:11:51.563620958 -0400
@@ -101,14 +101,35 @@
            int n = 0;
            int lines = listLines[0]->size();

+   qreal QPLengthSquared[lines];
+   qreal QPLength[lines];
+   qreal Q2P2Length[lines];
+
+   for(int k=0; k<lines; ++k) {
+                       // get original lines from reference line
+                       QPoint P = listLines[h]->at(k).first;
+                       QPoint Q = listLines[h]->at(k).second;
+
+       QVector2D QP(Q - P);
+       QPLength[k] = QP.length();
+       QPLengthSquared[k] = QP.lengthSquared();
+
+       // get interpolating lines from reference line
+       QPoint P2 = listAux[h]->at(k).first;
+       QPoint Q2 = listAux[h]->at(k).second;
+
+       QVector2D Q2P2(Q2 - P2);
```

```
+       Q2P2Length[k] = Q2P2.length();
+   }
+
            for(int i=0; i<wimg; ++i) {
                        for(int j=0; j<himg; ++j) {

                                    QPoint X(i, j);
                                    double u, v;

-                                   double ww[lines];
-                                   QPoint pp[lines];
+           QPoint sum(0.0, 0.0);
+           double wsum = 0;

                                    // for each line
                                    for(int k=0; k<lines; ++k) {
@@ -123,8 +144,8 @@
                                                QVector2D pQP(QP.y(), -QP.x());

                                                // Calculate u, v
-                                               u = QVector2D::dotProduct(XP, QP) /  QP.lengthSquared();
-                                               v = QVector2D::dotProduct(XP, pQP) / QP.length();
+                                               u = QVector2D::dotProduct(XP, QP) /  QPLengthSquared[k];
+                                               v = QVector2D::dotProduct(XP, pQP) / QPLength[k];

            // get interpolating lines from reference line
            QPoint P2 = listAux[h]->at(k).first;
@@ -133,7 +154,7 @@
            QVector2D Q2P2(Q2 - P2);
            QVector2D pQ2P2(Q2P2.y(), -Q2P2.x());

-           QVector2D X2 = QVector2D(P2) + u * Q2P2 + (v * pQ2P2) / Q2P2.length();
+           QVector2D X2 = QVector2D(P2) + u * Q2P2 + (v * pQ2P2) / Q2P2Length[k];

            QPoint p = X2.toPoint() - X;

@@ -143,20 +164,14 @@
            else dist = sqrt(pow(X.x() - Q.x(), 2.0) + pow(X.y() - Q.y(), 2.0));

            double w = 0;
-           w =  pow(QP.length(), VARP);
+           w =  pow(QPLength[k], VARP);
            w /= (VARA + dist);
            w = pow(w, VARB);

-           ww[k] = w;
-           pp[k] = p;
+           sum  += w * p;
+           wsum += w;
        }

-       QPoint sum(0.0, 0.0);
-       double wsum = 0;
-       for(int k=0; k<lines; ++k) {
-           sum  += ww[k] * pp[k];
-           wsum += ww[k];
-       }
        sum /= wsum;

        QPoint X2 = X + sum;
```

Performance Profiling:

Original run time: **25.550951754 seconds**
Original line-by-line profiling:

```
       |                    // Calculate u, v
       |                    u = QVector2D::dotProduct(XP, QP) /  QP.lengthSquared();
  0.42 |       divsd  %xmm0,%xmm1
 11.91 |       movsd  %xmm1,-0x78(%rbp)
```

Runtime with Improvement 1:

```
 Performance counter stats for './test_harness':

    23474.339171 task-clock              #    0.999 CPUs utilized
         2,380 context-switches          #    0.101 K/sec
             0 cpu-migrations            #    0.000 K/sec
         4,324 page-faults               #    0.184 K/sec
    88,937,844,317 cycles                #    3.789 GHz                      [83.33%]
    49,591,204,580 stalled-cycles-frontend   #   55.76% frontend cycles idle    [83.35%]
    20,724,239,275 stalled-cycles-backend    #   23.30% backend  cycles idle    [66.66%]
   116,186,984,702 instructions          #    1.31  insns per cycle
                                         #    0.43  stalled cycles per insn [83.33%]
    15,071,372,603 branches              #  642.036 M/sec                    [83.35%]
       359,444,986 branch-misses         #    2.38% of all branches          [83.32%]

    23.505277297 seconds time elapsed
```

Additional runs produced:

```
23.822664304 seconds time elapsed
23.563770868 seconds time elapsed
23.465816740 seconds time elapsed
23.453422653 seconds time elapsed
23.461707053 seconds time elapsed
```

average runtime: **23.54544315 seconds**

Line-by-line profiling with Improvement 1:

```
       |                                    u = QVector2D::dotProduct(XP, QP) /  QPLengthSquared[k];
  0.39 |       divsd  (%rdx,%rbx,8),%xmm0
  9.72 |       movsd  %xmm0,-0x68(%rbp)
```

Discussion:

Improvement 1 increased performance by a factor of 1.09 (9% speed increase). The first part of the change involved moving the sum calculation into the innermost for loop (loop through 0 – k). In the original code, the innermost loop iterates k times, and the values of w and p are stored for each iteration. The arrays of w's and p's from those k iterations are then used to compute the sum and wsum variables. The improvement updates the sum and wsum after each iteration in the innermost loop, thus the performance gain comes from a reduced number of instructions because we no longer need to store the w and p values into temporary storage and later retrieve those values. The fundamental operations of the program is unchanged because the calculations of sum and wsum are the same, whether we get the values for w and p for k iterations then summing over w * p for each iteration, or updating the sum at the end of each iteration immediately. The second part of the change involved pre-computing the values of length() for vector QP and Q2P2, and lengthSquared() for QP. From the Base Profile, we found that those 2 methods calls take up a significant percentage of the total runtime. It is noted that the methods are called i * j * k times. However, we notice that QP and Q2P2 are vectors that are not dependent on the outer loop variables i and j. That is, they are invariant with respect to i and j. Thus, we

can take those calculations outside the outer loops of i and j and compute them k times only (because QP and Q2P2 depend on k), which is a big performance win for us! Of course, the fundamental operations of the program is unchanged because there are only k different values for QP and Q2P2, so computing those corresponding method calls() outside the i and j loops then using them in the k loop yields the same result as recomputing those method call values for every single iteration of the triple for-loop.

Improvement 2: OpenMP (this was done on top of Improvement 1)

Code diff:

```
--- model_intermediate.cpp   2013-03-24 01:11:51.563620958 -0400
+++ model_final.cpp          2013-03-24 01:32:40.200373404 -0400
@@ -105,6 +105,7 @@
    qreal QPLength[lines];
    qreal Q2P2Length[lines];

+   #pragma omp parallel
    for(int k=0; k<lines; ++k) {
                        // get original lines from reference line
                        QPoint P = listLines[h]->at(k).first;
@@ -122,6 +123,7 @@
        Q2P2Length[k] = Q2P2.length();
    }

+   #pragma omp parallel for collapse (2)
            for(int i=0; i<wimg; ++i) {
                        for(int j=0; j<himg; ++j) {
```

Performance Profiling:

We used 4 threads on ece459-1.uwaterloo.ca (we exported OMP_NUM_THREADS=4).

Runtime with Improvement 1: **23.54544315 seconds**

Runtime with Improvement 2:

```
 Performance counter stats for './test_harness':

     32389.436273 task-clock                #    3.215 CPUs utilized
          3,352 context-switches          #    0.103 K/sec
             8 cpu-migrations            #    0.000 K/sec
          4,982 page-faults               #    0.154 K/sec
   115,552,504,732 cycles                    #    3.568 GHz                     [83.34%]
    74,938,815,697 stalled-cycles-frontend   #   64.85% frontend cycles idle    [83.32%]
    32,919,916,709 stalled-cycles-backend    #   28.49% backend  cycles idle    [66.71%]
   117,935,927,606 instructions              #    1.02  insns per cycle
                                  #    0.64  stalled cycles per insn [83.37%]
    15,151,904,139 branches                  #  467.804 M/sec                   [83.32%]
       380,757,859 branch-misses             #    2.51% of all branches         [83.32%]

     10.074092328 seconds time elapsed
```

Additional runs produced:

```
10.087843307 seconds time elapsed
10.134689924 seconds time elapsed
10.112949016 seconds time elapsed
10.054258634 seconds time elapsed
10.148231980 seconds time elapsed
```

average runtime: **10.10201086 seconds**

Discussion:

Improvement 2 increased performance by a factor of 2.33 (133% speed increase). By collapsing the loops i and j into a single loop, and parallelizing it using a team of threads, we get a huge performance boost. We get great performance because the threads do their work in parallel, allowing for more work to be done than sequentially with a single process. Having threads split the work and doing them in parallel does not change the fundamental operation of the program because each iteration of i, j is independent of other iterations of i, j. Thus work done by each thread will not impact any work performed by another thread.

Appendix

Output of *perf annotate*:

```
 Percent |       Source code & Disassembly of test_harness
------------------------------------------------
         :
         :
         :
         :       Disassembly of section .text:
         :
         :       0000000000401b80 <Model::morph(int, double, double, double)>:
         :                   }
         :               }
         :           }
         :       }
         :
         :       void Model::morph(int h, double VARA, double VARB, double VARP) {
    0.00 :         401b80:       push   %rbp
    0.00 :         401b81:       mov    %rsp,%rbp
    0.00 :         401b84:       push   %r15
    0.00 :         401b86:       push   %r14
    0.00 :         401b88:       mov    %rdi,%r14
    0.00 :         401b8b:       push   %r13
    0.00 :         401b8d:       push   %r12
    0.00 :         401b8f:       push   %rbx
    0.00 :         401b90:       sub    $0x118,%rsp
         :           int n = 0;
         :           int lines = listLines[0]->size();
    0.00 :         401b97:       mov    0x8(%rdi),%rax
         :
         :           for(int i=0; i<wimg; ++i) {
    0.00 :         401b9b:       mov    0x4(%rdi),%r9d
         :                   }
         :               }
         :           }
         :       }
         :
         :       void Model::morph(int h, double VARA, double VARB, double VARP) {
    0.00 :         401b9f:       movsd  %xmm0,-0xc8(%rbp)
    0.00 :         401ba7:       movsd  %xmm1,-0xd0(%rbp)
         :           int n = 0;
         :           int lines = listLines[0]->size();
    0.00 :         401baf:       mov    (%rax),%rdx
         :                   }
         :               }
         :           }
         :       }
         :
         :       void Model::morph(int h, double VARA, double VARB, double VARP) {
    0.00 :         401bb2:       movsd  %xmm2,-0xd8(%rbp)
         :
         :            // [23.2.4.2] capacity
         :            /**  Returns the number of elements in the %vector.  */
         :            size_type
         :            size() const _GLIBCXX_NOEXCEPT
         :            { return size_type(this->_M_impl._M_finish - this->_M_impl._M_start); }
    0.00 :         401bba:       mov    0x8(%rdx),%rax
    0.00 :         401bbe:       sub    (%rdx),%rax
    0.00 :         401bc1:       sar    $0x4,%rax
         :           int n = 0;
         :           int lines = listLines[0]->size();
         :
         :           for(int i=0; i<wimg; ++i) {
```

```
 0.00 :          401bc5:       test   %r9d,%r9d
      :            }
      :        }
      :
      :        void Model::morph(int h, double VARA, double VARB, double VARP) {
      :            int n = 0;
      :            int lines = listLines[0]->size();
 0.00 :          401bc8:       mov    %eax,-0xe4(%rbp)
      :
      :            for(int i=0; i<wimg; ++i) {
 0.00 :          401bce:       jle    402352 <Model::morph(int, double, double, double)+0x7d2>
 0.00 :          401bd4:       movslq %eax,%rdx
      :                    }
      :                }
      :            }
      :        }
      :
      :        void Model::morph(int h, double VARA, double VARB, double VARP) {
 0.00 :          401bd7:       sub    $0x1,%eax
      :            int n = 0;
      :            int lines = listLines[0]->size();
      :
      :            for(int i=0; i<wimg; ++i) {
 0.00 :          401bda:       movl   $0x0,-0x98(%rbp)
 0.00 :          401be4:       mov    %rdx,-0x110(%rbp)
 0.00 :          401beb:       mov    -0x110(%rbp),%rcx
 0.00 :          401bf2:       sub    $0x1,%rdx
 0.00 :          401bf6:       mov    %rdx,-0x108(%rbp)
      :
      :                    // for each line
      :                    for(int k=0; k<lines; ++k) {
      :
      :                        // get original lines from reference line
      :                        QPoint P = listLines[h]->at(k).first;
 0.00 :          401bfd:       movslq %esi,%rdx
      :                    if(y0 >= himg) y0 = himg-1;
      :
      :                    X2 = QPoint(x0, y0);
      :
      :                    if(X2 == X) n++;
      :                    imgs[h+2]->setPixel(X, imgs[h]->pixel(X2));
 0.00 :          401c00:       add    $0x2,%esi
      :
      :                    // for each line
      :                    for(int k=0; k<lines; ++k) {
      :
      :                        // get original lines from reference line
      :                        QPoint P = listLines[h]->at(k).first;
 0.00 :          401c03:       mov    %rdx,-0x118(%rbp)
      :                    if(y0 >= himg) y0 = himg-1;
      :
      :                    X2 = QPoint(x0, y0);
      :
      :                    if(X2 == X) n++;
      :                    imgs[h+2]->setPixel(X, imgs[h]->pixel(X2));
 0.00 :          401c0a:       movslq %esi,%rsi
      :
      :                    // for each line
      :                    for(int k=0; k<lines; ++k) {
      :
      :                        // get original lines from reference line
      :                        QPoint P = listLines[h]->at(k).first;
 0.00 :          401c0d:       shl    $0x3,%rdx
 0.00 :          401c11:       shl    $0x3,%rcx
```

```
 0.00 :          401c15:       mov    %rdx,-0xb0(%rbp)
      :                     if(y0 >= himg) y0 = himg-1;
      :
      :                     X2 = QPoint(x0, y0);
      :
      :                     if(X2 == X) n++;
      :                     imgs[h+2]->setPixel(X, imgs[h]->pixel(X2));
 0.00 :          401c1c:       mov    %rsi,-0x120(%rbp)
 0.00 :          401c23:       mov    %rcx,-0x128(%rbp)
      :                     }
      :                 }
      :               }
      :             }
      :
      :             void Model::morph(int h, double VARA, double VARB, double VARP) {
 0.00 :          401c2a:       mov    %rax,-0xe0(%rbp)
      :               int n = 0;
      :               int lines = listLines[0]->size();
      :
      :               for(int i=0; i<wimg; ++i) {
      :                   for(int j=0; j<himg; ++j) {
 0.00 :          401c31:       mov    (%r14),%edi
 0.00 :          401c34:       test   %edi,%edi
 0.00 :          401c36:       jle    40233b <Model::morph(int, double, double, double)+0x7bb>
 0.00 :          401c3c:       mov    -0x128(%rbp),%rcx
 0.00 :          401c43:       mov    -0x128(%rbp),%rax
 0.00 :          401c4a:       movl   $0x0,-0x94(%rbp)
 0.00 :          401c54:       add    $0x16,%rcx
 0.00 :          401c58:       add    $0x12,%rax
      :
      :                     QPoint X(i, j);
      :                     double u, v;
      :
      :                     double ww[lines];
 0.00 :          401c5c:       and    $0xfffffffffffffff0,%rcx
      :                     QPoint pp[lines];
 0.00 :          401c60:       and    $0xfffffffffffffff0,%rax
      :                   for(int j=0; j<himg; ++j) {
      :
      :                     QPoint X(i, j);
      :                     double u, v;
      :
      :                     double ww[lines];
 0.00 :          401c64:       mov    %rcx,-0xf8(%rbp)
      :                     QPoint pp[lines];
 0.00 :          401c6b:       mov    %rax,-0x100(%rbp)
 0.00 :          401c72:       nopw   0x0(%rax,%rax,1)
 0.10 :          401c78:       mov    %rsp,-0xf0(%rbp)
      :                   for(int j=0; j<himg; ++j) {
      :
      :                     QPoint X(i, j);
      :                     double u, v;
      :
      :                     double ww[lines];
 0.02 :          401c7f:       sub    -0xf8(%rbp),%rsp
      :                     }
      :                   }
      :               }
      :             }
      :
      :             void Model::morph(int h, double VARA, double VARB, double VARP) {
 0.01 :          401c86:       xor    %eax,%eax
      :                   for(int j=0; j<himg; ++j) {
      :
```

```
      :                    QPoint X(i, j);
      :                    double u, v;
      :
      :                    double ww[lines];
 0.03 :      401c88:       mov    %rsp,-0xc0(%rbp)
      :                    QPoint pp[lines];
 0.08 :      401c8f:       sub    -0x100(%rbp),%rsp
 0.05 :      401c96:       cmpq   $0xffffffffffffffff,-0x108(%rbp)
 0.00 :      401c9e:       mov    %rsp,-0xb8(%rbp)
 0.01 :      401ca5:       je     401cd0 <Model::morph(int, double, double, double)+0x150>
 0.09 :      401ca7:       mov    -0xb8(%rbp),%rdx
 0.10 :      401cae:       mov    -0x110(%rbp),%rcx
 0.00 :      401cb5:       nopl   (%rax)
      :                  /*****************************************************************************
      :                    QPoint inline functions
      :                   *****************************************************************************/
      :
      :                  inline QPoint::QPoint()
      :                  { xp=0; yp=0; }
 0.12 :      401cb8:       movl   $0x0,(%rdx,%rax,8)
 0.45 :      401cbf:       movl   $0x0,0x4(%rdx,%rax,8)
 0.29 :      401cc7:       add    $0x1,%rax
 0.11 :      401ccb:       cmp    %rcx,%rax
 0.00 :      401cce:       jne    401cb8 <Model::morph(int, double, double, double)+0x138>
      :
      :                    // for each line
      :                    for(int k=0; k<lines; ++k) {
 0.05 :      401cd0:       mov    -0xe4(%rbp),%r8d
 0.06 :      401cd7:       test   %r8d,%r8d
 0.00 :      401cda:       jle    4023da <Model::morph(int, double, double, double)+0x85a>
      :
      :                        // get original lines from reference line
      :                        QPoint P = listLines[h]->at(k).first;
 0.02 :      401ce0:       mov    0x8(%r14),%rax
 0.00 :      401ce4:       mov    -0xb0(%rbp),%rcx
 0.04 :      401ceb:       mov    (%rax,%rcx,1),%rdx
      :                  public:
      :                      virtual void raise() const;
      :                      virtual Exception *clone() const;
      :                  };
      :
      :                  class Q_CORE_EXPORT UnhandledException : public Exception
 0.06 :      401cef:       mov    (%rdx),%rax
 0.07 :      401cf2:       mov    0x8(%rdx),%rdx
 0.01 :      401cf6:       sub    %rax,%rdx
 0.07 :      401cf9:       sar    $0x4,%rdx
      :                    protected:
      :                      /// Safety check used only from at().
      :                      void
      :                      _M_range_check(size_type __n) const
      :                      {
      :                        if (__n >= this->size())
 0.07 :      401cfd:       test   %rdx,%rdx
 0.03 :      401d00:       je     4023d0 <Model::morph(int, double, double, double)+0x850>
 0.07 :      401d06:       mov    (%rax),%ecx
 0.04 :      401d08:       mov    0x4(%rax),%edx
      :                    }
      :                  }
      :                }
      :
      :                void Model::morph(int h, double VARA, double VARB, double VARP) {
 0.02 :      401d0b:       xor    %ebx,%ebx
      :                   *  out_of_range lookups are not defined. (For checked lookups
```

```
        :                * see at().)
        :                */
        :               reference
        :               operator[](size_type __n)
        :               { return *(this->_M_impl._M_start + __n); }
   0.03 :         401d0d:       xor    %r12d,%r12d
        :
        :                     // for each line
        :                     for(int k=0; k<lines; ++k) {
        :
        :                         // get original lines from reference line
        :                         QPoint P = listLines[h]->at(k).first;
   0.04 :         401d10:       xor    %r13d,%r13d
   0.01 :         401d13:       jmpq   401e72 <Model::morph(int, double, double, double)+0x2f2>
   0.00 :         401d18:       nopl   0x0(%rax,%rax,1)
        :           return qFuzzyCompare(v1.xp, v2.xp) && qFuzzyCompare(v1.yp, v2.yp);
        :       }
        :
        :       inline QPoint QVector2D::toPoint() const
        :       {
        :           return QPoint(qRound(xp), qRound(yp));
   0.46 :         401d20:       unpcklps %xmm1,%xmm1
        :
        :       template <typename T>
        :       Q_DECL_CONSTEXPR inline T qAbs(const T &t) { return t >= 0 ? t : -t; }
        :
        :       Q_DECL_CONSTEXPR inline int qRound(qreal d)
        :       { return d >= qreal(0.0) ? int(d + qreal(0.5)) : int(d - int(d-1) + qreal(0.5)) + int(d-1); }
   0.05 :         401d23:       movsd  0xbe5(%rip),%xmm3        # 402910 <_IO_stdin_used+0x60>
   0.02 :         401d2b:       cvtps2pd %xmm1,%xmm1
   0.43 :         401d2e:       addsd  %xmm3,%xmm0
   0.01 :         401d32:       movsd  %xmm3,-0x80(%rbp)
   0.00 :         401d37:       xorpd  %xmm3,%xmm3
   0.00 :         401d3b:       cvttsd2si %xmm0,%r12d
   1.79 :         401d40:       ucomisd %xmm3,%xmm1
   0.00 :         401d44:       jb     4020ce <Model::morph(int, double, double, double)+0x54e>
   0.04 :         401d4a:       addsd  -0x80(%rbp),%xmm1
   0.51 :         401d4f:       cvttsd2si %xmm1,%r13d
        :                     QVector2D X2 = QVector2D(P2) + u * Q2P2 + (v * pQ2P2) / Q2P2.length();
        :
        :                     QPoint p = X2.toPoint() - X;
        :
        :                     double dist = 0;
        :                     if(u > 0 && u < 1) dist = fabs(v);
   0.03 :         401d54:       xorpd  %xmm1,%xmm1
        :
        :       inline const QPoint operator+(const QPoint &p1, const QPoint &p2)
        :       { return QPoint(p1.xp+p2.xp, p1.yp+p2.yp); }
        :
        :       inline const QPoint operator-(const QPoint &p1, const QPoint &p2)
        :       { return QPoint(p1.xp-p2.xp, p1.yp-p2.yp); }
   0.05 :         401d58:       sub    -0x94(%rbp),%r12d
   0.47 :         401d5f:       movsd  -0x78(%rbp),%xmm0
   0.00 :         401d64:       sub    -0x98(%rbp),%r13d
   0.00 :         401d6b:       ucomisd %xmm1,%xmm0
   0.44 :         401d6f:       jbe    401d84 <Model::morph(int, double, double, double)+0x204>
   0.00 :         401d71:       movsd  0xbdf(%rip),%xmm0        # 402958 <_IO_stdin_used+0xa8>
   0.01 :         401d79:       ucomisd -0x78(%rbp),%xmm0
   0.38 :         401d7e:       ja     402118 <Model::morph(int, double, double, double)+0x598>
        :                     else if(u <= 0) dist = sqrt(pow(X.x() - P.x(), 2.0) + pow(X.y() - P.y(), 2.0));
   0.01 :         401d84:       xorpd  %xmm3,%xmm3
   0.00 :         401d88:       ucomisd -0x78(%rbp),%xmm3
   0.23 :         401d8d:       jae    402100 <Model::morph(int, double, double, double)+0x580>
        :                     else dist = sqrt(pow(X.x() - Q.x(), 2.0) + pow(X.y() - Q.y(), 2.0));
```

```
 0.00 :          401d93:       mov    -0x98(%rbp),%eax
 0.00 :          401d99:       sub    %r15d,%eax
 0.01 :          401d9c:       cvtsi2sd %eax,%xmm1
 0.11 :          401da0:       mov    -0x94(%rbp),%eax
 0.00 :          401da6:       sub    -0x9c(%rbp),%eax
 0.01 :          401dac:       cvtsi2sd %eax,%xmm0
 0.11 :          401db0:       mulsd  %xmm0,%xmm0
 0.01 :          401db4:       mulsd  %xmm1,%xmm1
 0.13 :          401db8:       addsd  %xmm0,%xmm1
 0.11 :          401dbc:       sqrtsd %xmm1,%xmm0
 0.64 :          401dc0:       ucomisd %xmm0,%xmm0
 0.18 :          401dc4:       jp     4023b8 <Model::morph(int, double, double, double)+0x838>
 0.10 :          401dca:       movsd  %xmm0,-0x78(%rbp)
      :
      :                         double w = 0;
      :                         w =  pow(QP.length(), VARP);
 0.00 :          401dcf:       lea    -0x60(%rbp),%rdi
 0.02 :          401dd3:       callq  400e20 <QVector2D::length() const@plt>
 0.00 :          401dd8:       movsd  -0xd8(%rbp),%xmm1
 0.00 :          401de0:       callq  400e90 <pow@plt>
      :                         w /= (VARA + dist);
 0.49 :          401de5:       movsd  -0xc8(%rbp),%xmm1
 0.00 :          401ded:       addsd  -0x78(%rbp),%xmm1
 0.00 :          401df2:       divsd  %xmm1,%xmm0
      :                         w = pow(w, VARB);
 0.85 :          401df6:       movsd  -0xd0(%rbp),%xmm1
 0.48 :          401dfe:       callq  400e90 <pow@plt>
      :
      :                         ww[k] = w;
 0.53 :          401e03:       mov    -0xc0(%rbp),%rax
      :                         pp[k] = p;
 0.00 :          401e0a:       mov    -0xb8(%rbp),%rdx
      :
      :                     double ww[lines];
      :                     QPoint pp[lines];
      :
      :                     // for each line
      :                     for(int k=0; k<lines; ++k) {
 0.00 :          401e11:       cmp    -0xe0(%rbp),%rbx
      :                         double w = 0;
      :                         w =  pow(QP.length(), VARP);
      :                         w /= (VARA + dist);
      :                         w = pow(w, VARB);
      :
      :                         ww[k] = w;
 0.00 :          401e18:       movsd  %xmm0,(%rax,%rbx,8)
      :                         pp[k] = p;
 0.46 :          401e1d:       movslq -0xa8(%rbp),%rax
 0.01 :          401e24:       lea    (%rdx,%rax,8),%rax
 0.00 :          401e28:       mov    %r13d,(%rax)
 0.66 :          401e2b:       mov    %r12d,0x4(%rax)
      :
      :                     double ww[lines];
      :                     QPoint pp[lines];
      :
      :                     // for each line
      :                     for(int k=0; k<lines; ++k) {
 0.07 :          401e2f:       je     402140 <Model::morph(int, double, double, double)+0x5c0>
      :
      :                         // get original lines from reference line
      :                         QPoint P = listLines[h]->at(k).first;
 0.00 :          401e35:       mov    -0xb0(%rbp),%rdx
 0.00 :          401e3c:       mov    0x8(%r14),%rax
      :                     }
```

```
        :                }
        :            }
        :        }
        :
        :        void Model::morph(int h, double VARA, double VARB, double VARP) {
   0.44 :          401e40:       add    $0x1,%rbx
        :
        :                    // for each line
        :                    for(int k=0; k<lines; ++k) {
        :
        :                        // get original lines from reference line
        :                        QPoint P = listLines[h]->at(k).first;
   0.01 :          401e44:       mov    (%rax,%rdx,1),%rax
   0.02 :          401e48:       mov    (%rax),%rdx
        :
        :                // [23.2.4.2] capacity
        :                /**  Returns the number of elements in the %vector.  */
        :                size_type
        :                size() const _GLIBCXX_NOEXCEPT
        :                { return size_type(this->_M_impl._M_finish - this->_M_impl._M_start); }
   0.05 :          401e4b:       mov    0x8(%rax),%rax
   0.27 :          401e4f:       sub    %rdx,%rax
   0.03 :          401e52:       sar    $0x4,%rax
        :              protected:
        :                /// Safety check used only from at().
        :                void
        :                _M_range_check(size_type __n) const
        :                {
        :                  if (__n >= this->size())
   0.00 :          401e56:       cmp    %rax,%rbx
   0.00 :          401e59:       jae    4023d0 <Model::morph(int, double, double, double)+0x850>
        :                 *  out_of_range lookups are not defined. (For checked lookups
        :                 *  see at().)
        :                 */
        :                reference
        :                operator[](size_type __n)
        :                { return *(this->_M_impl._M_start + __n); }
   0.01 :          401e5f:       mov    %rbx,%r12
   0.35 :          401e62:       mov    %rbx,%r13
   0.02 :          401e65:       shl    $0x4,%r12
   0.00 :          401e69:       lea    (%rdx,%r12,1),%rax
   0.01 :          401e6d:       mov    (%rax),%ecx
   0.31 :          401e6f:       mov    0x4(%rax),%edx
        :                        QPoint Q = listLines[h]->at(k).second;
   0.05 :          401e72:       mov    0x8(%rax),%r15d
   0.08 :          401e76:       mov    0xc(%rax),%eax
        :                        QVector2D QP(Q - P);
        :
        :                        QVector2D pQP(QP.y(), -QP.x());
        :
        :                        // Calculate u, v
        :                        u = QVector2D::dotProduct(XP, QP) /  QP.lengthSquared();
   0.06 :          401e79:       lea    -0x60(%rbp),%rsi
   0.35 :          401e7d:       lea    -0x70(%rbp),%rdi
   0.05 :          401e81:       mov    %ebx,-0xa8(%rbp)
        :                    // for each line
        :                    for(int k=0; k<lines; ++k) {
        :
        :                        // get original lines from reference line
        :                        QPoint P = listLines[h]->at(k).first;
        :                        QPoint Q = listLines[h]->at(k).second;
   0.07 :          401e87:       mov    %eax,-0x9c(%rbp)
   0.06 :          401e8d:       mov    -0x94(%rbp),%eax
   0.35 :          401e93:       sub    %edx,%eax
```

```
 0.03 :          401e95:       mov    %eax,-0xa0(%rbp)
 0.08 :          401e9b:       mov    -0x98(%rbp),%eax
 0.04 :          401ea1:       sub    %ecx,%eax
      :
      :        inline QVector2D::QVector2D(float xpos, float ypos, int) : xp(xpos), yp(ypos) {}
      :
      :        inline QVector2D::QVector2D(qreal xpos, qreal ypos) : xp(xpos), yp(ypos) {}
      :
      :        inline QVector2D::QVector2D(const QPoint& point) : xp(point.x()), yp(point.y()) {}
 0.38 :          401ea3:       cvtsi2ss %eax,%xmm0
 0.33 :          401ea7:       mov    %eax,-0xa4(%rbp)
 0.02 :          401ead:       mov    %r15d,%eax
 0.23 :          401eb0:       sub    %ecx,%eax
 0.03 :          401eb2:       movss  %xmm0,-0x70(%rbp)
 0.12 :          401eb7:       cvtsi2ssl -0xa0(%rbp),%xmm0
 0.29 :          401ebf:       movss  %xmm0,-0x6c(%rbp)
 0.35 :          401ec4:       cvtsi2ss %eax,%xmm0
 0.21 :          401ec8:       mov    -0x9c(%rbp),%eax
 0.01 :          401ece:       sub    %edx,%eax
 0.37 :          401ed0:       cvtsi2ss %eax,%xmm1
 0.20 :          401ed4:       movss  %xmm0,-0x60(%rbp)
      :        inline bool QVector2D::isNull() const
      :        {
      :            return qIsNull(xp) && qIsNull(yp);
      :        }
      :
      :        inline qreal QVector2D::x() const { return qreal(xp); }
 0.00 :          401ed9:       unpcklps %xmm0,%xmm0
 0.32 :          401edc:       cvtps2pd %xmm0,%xmm0
      :
      :        inline QVector2D::QVector2D(float xpos, float ypos, int) : xp(xpos), yp(ypos) {}
      :
      :        inline QVector2D::QVector2D(qreal xpos, qreal ypos) : xp(xpos), yp(ypos) {}
      :
      :        inline QVector2D::QVector2D(const QPoint& point) : xp(point.x()), yp(point.y()) {}
 0.32 :          401edf:       movss  %xmm1,-0x5c(%rbp)
      :
      :                        QVector2D XP(X - P);
      :                        QVector2D QP(Q - P);
      :
      :                        QVector2D pQP(QP.y(), -QP.x());
 0.02 :          401ee4:       xorpd  0xa54(%rip),%xmm0        # 402940 <_IO_stdin_used+0x90>
      :
      :        inline QVector2D::QVector2D() : xp(0.0f), yp(0.0f) {}
      :
      :        inline QVector2D::QVector2D(float xpos, float ypos, int) : xp(xpos), yp(ypos) {}
      :
      :        inline QVector2D::QVector2D(qreal xpos, qreal ypos) : xp(xpos), yp(ypos) {}
 0.37 :          401eec:       movss  %xmm1,-0x50(%rbp)
 0.04 :          401ef1:       unpcklpd %xmm0,%xmm0
 0.14 :          401ef5:       cvtpd2ps %xmm0,%xmm0
 0.79 :          401ef9:       movss  %xmm0,-0x4c(%rbp)
      :
      :                        // Calculate u, v
      :                        u = QVector2D::dotProduct(XP, QP) /  QP.lengthSquared();
 0.15 :          401efe:       callq  400ed0 <QVector2D::dotProduct(QVector2D const&, QVector2D const&)@plt>
 0.00 :          401f03:       lea    -0x60(%rbp),%rdi
 0.00 :          401f07:       movsd  %xmm0,-0x78(%rbp)
 0.37 :          401f0c:       callq  400f40 <QVector2D::lengthSquared() const@plt>
 0.00 :          401f11:       movsd  -0x78(%rbp),%xmm1
      :                        v = QVector2D::dotProduct(XP, pQP) / QP.length();
 0.06 :          401f16:       lea    -0x50(%rbp),%rsi
 0.03 :          401f1a:       lea    -0x70(%rbp),%rdi
      :                        QVector2D QP(Q - P);
```

```
       :
       :                        QVector2D pQP(QP.y(), -QP.x());
       :
       :                        // Calculate u, v
       :                        u = QVector2D::dotProduct(XP, QP) /  QP.lengthSquared();
  0.42 :        401f1e:       divsd  %xmm0,%xmm1
 11.91 :        401f22:       movsd  %xmm1,-0x78(%rbp)
       :                        v = QVector2D::dotProduct(XP, pQP) / QP.length();
  0.39 :        401f27:       callq  400ed0 <QVector2D::dotProduct(QVector2D const&, QVector2D const&)@plt>
  0.00 :        401f2c:       lea    -0x60(%rbp),%rdi
  0.00 :        401f30:       movsd  %xmm0,-0x88(%rbp)
  0.07 :        401f38:       callq  400e20 <QVector2D::length() const@plt>
  0.00 :        401f3d:       movsd  -0x88(%rbp),%xmm2
       :
       :                        // get interpolating lines from reference line
       :                        QPoint P2 = listAux[h]->at(k).first;
  0.00 :        401f45:       mov    0x10(%r14),%rax
  0.02 :        401f49:       mov    -0xb0(%rbp),%rcx
       :
       :                        QVector2D pQP(QP.y(), -QP.x());
       :
       :                        // Calculate u, v
       :                        u = QVector2D::dotProduct(XP, QP) /  QP.lengthSquared();
       :                        v = QVector2D::dotProduct(XP, pQP) / QP.length();
  0.49 :        401f50:       divsd  %xmm0,%xmm2
       :
       :                        // get interpolating lines from reference line
       :                        QPoint P2 = listAux[h]->at(k).first;
  7.73 :        401f54:       mov    (%rax,%rcx,1),%rdx
  0.00 :        401f58:       mov    (%rdx),%rcx
       :
       :            // [23.2.4.2] capacity
       :            /**  Returns the number of elements in the %vector.  */
       :            size_type
       :            size() const _GLIBCXX_NOEXCEPT
       :            { return size_type(this->_M_impl._M_finish - this->_M_impl._M_start); }
  0.00 :        401f5b:       mov    0x8(%rdx),%rax
  0.00 :        401f5f:       sub    %rcx,%rax
  0.47 :        401f62:       sar    $0x4,%rax
       :          protected:
       :            /// Safety check used only from at().
       :            void
       :            _M_range_check(size_type __n) const
       :            {
       :              if (__n >= this->size())
  0.00 :        401f66:       cmp    %r13,%rax
       :
       :                        QVector2D pQP(QP.y(), -QP.x());
       :
       :                        // Calculate u, v
       :                        u = QVector2D::dotProduct(XP, QP) /  QP.lengthSquared();
       :                        v = QVector2D::dotProduct(XP, pQP) / QP.length();
  0.00 :        401f69:       movsd  %xmm2,-0x88(%rbp)
  0.00 :        401f71:       jbe    4023c6 <Model::morph(int, double, double, double)+0x846>
       :             *  out_of_range lookups are not defined. (For checked lookups
       :             *  see at().)
       :             */
       :            reference
       :            operator[](size_type __n)
       :            { return *(this->_M_impl._M_start + __n); }
  0.41 :        401f77:       lea    (%rcx,%r12,1),%rax
       :                        QPoint Q2 = listAux[h]->at(k).second;
       :
       :                        QVector2D Q2P2(Q2 - P2);
```

```
      :                         QVector2D pQ2P2(Q2P2.y(), -Q2P2.x());
      :
      :                         QVector2D X2 = QVector2D(P2) + u * Q2P2 + (v * pQ2P2) / Q2P2.length();
 0.00 :          401f7b:       lea    -0x40(%rbp),%rdi
      :                         // Calculate u, v
      :                         u = QVector2D::dotProduct(XP, QP) /  QP.lengthSquared();
      :                         v = QVector2D::dotProduct(XP, pQP) / QP.length();
      :
      :                         // get interpolating lines from reference line
      :                         QPoint P2 = listAux[h]->at(k).first;
 0.00 :          401f7f:       mov    (%rax),%r13d
 0.00 :          401f82:       mov    0x4(%rax),%r12d
 0.43 :          401f86:       mov    0xc(%rax),%edx
 0.00 :          401f89:       mov    0x8(%rax),%eax
 0.00 :          401f8c:       sub    %r13d,%eax
 0.00 :          401f8f:       sub    %r12d,%edx
      :
      :        inline QVector2D::QVector2D(const QPoint& point) : xp(point.x()), yp(point.y()) {}
 0.49 :          401f92:       cvtsi2ss %eax,%xmm0
 0.00 :          401f96:       cvtsi2ss %edx,%xmm1
 0.47 :          401f9a:       movss  %xmm0,-0x40(%rbp)
      :        inline bool QVector2D::isNull() const
      :        {
      :            return qIsNull(xp) && qIsNull(yp);
      :        }
      :
      :        inline qreal QVector2D::x() const { return qreal(xp); }
 0.00 :          401f9f:       unpcklps %xmm0,%xmm0
      :
      :        inline QVector2D::QVector2D(float xpos, float ypos, int) : xp(xpos), yp(ypos) {}
      :
      :        inline QVector2D::QVector2D(qreal xpos, qreal ypos) : xp(xpos), yp(ypos) {}
      :
      :        inline QVector2D::QVector2D(const QPoint& point) : xp(point.x()), yp(point.y()) {}
 0.00 :          401fa2:       movss  %xmm1,-0x3c(%rbp)
      :        {
      :            return qIsNull(xp) && qIsNull(yp);
      :        }
      :
      :        inline qreal QVector2D::x() const { return qreal(xp); }
      :        inline qreal QVector2D::y() const { return qreal(yp); }
 0.00 :          401fa7:       unpcklps %xmm1,%xmm1
      :        inline bool QVector2D::isNull() const
      :        {
      :            return qIsNull(xp) && qIsNull(yp);
      :        }
      :
      :        inline qreal QVector2D::x() const { return qreal(xp); }
 0.54 :          401faa:       cvtps2pd %xmm0,%xmm0
      :        inline qreal QVector2D::y() const { return qreal(yp); }
 0.00 :          401fad:       cvtps2pd %xmm1,%xmm1
      :                         QPoint Q2 = listAux[h]->at(k).second;
      :
      :                         QVector2D Q2P2(Q2 - P2);
      :                         QVector2D pQ2P2(Q2P2.y(), -Q2P2.x());
 0.50 :          401fb0:       xorpd  0x988(%rip),%xmm0        # 402940 <_IO_stdin_used+0x90>
 0.00 :          401fb8:       movsd  %xmm1,-0x80(%rbp)
      :
      :        inline QVector2D::QVector2D() : xp(0.0f), yp(0.0f) {}
      :
      :        inline QVector2D::QVector2D(float xpos, float ypos, int) : xp(xpos), yp(ypos) {}
      :
      :        inline QVector2D::QVector2D(qreal xpos, qreal ypos) : xp(xpos), yp(ypos) {}
 0.00 :          401fbd:       unpcklpd %xmm0,%xmm0
```

```
 0.00 :          401fc1:       cvtpd2ps %xmm0,%xmm0
 0.48 :          401fc5:       movss  %xmm0,-0x90(%rbp)
      :
      :                        QVector2D X2 = QVector2D(P2) + u * Q2P2 + (v * pQ2P2) / Q2P2.length();
 0.00 :          401fcd:       callq  400e20 <QVector2D::length() const@plt>
      :            return QVector2D(v1.xp - v2.xp, v1.yp - v2.yp, 1);
      :        }
      :
      :        inline const QVector2D operator*(qreal factor, const QVector2D &vector)
      :        {
      :            return QVector2D(vector.xp * factor, vector.yp * factor, 1);
 0.00 :          401fd2:       movsd  -0x88(%rbp),%xmm1
 0.00 :          401fda:       movss  -0x40(%rbp),%xmm2
      :
      :        inline QVector2D::QVector2D(float xpos, float ypos, int) : xp(xpos), yp(ypos) {}
      :
      :        inline QVector2D::QVector2D(qreal xpos, qreal ypos) : xp(xpos), yp(ypos) {}
      :
      :        inline QVector2D::QVector2D(const QPoint& point) : xp(point.x()), yp(point.y()) {}
 0.01 :          401fdf:       cvtsi2ss %r13d,%xmm3
      :            return QVector2D(v1.xp - v2.xp, v1.yp - v2.yp, 1);
      :        }
      :
      :        inline const QVector2D operator*(qreal factor, const QVector2D &vector)
      :        {
      :            return QVector2D(vector.xp * factor, vector.yp * factor, 1);
 0.45 :          401fe4:       mulsd  -0x80(%rbp),%xmm1
 0.00 :          401fe9:       cvtps2pd %xmm2,%xmm2
 0.43 :          401fec:       mulsd  -0x78(%rbp),%xmm2
 0.00 :          401ff1:       unpcklpd %xmm1,%xmm1
 0.00 :          401ff5:       cvtpd2ps %xmm1,%xmm1
 0.54 :          401ff9:       unpcklpd %xmm2,%xmm2
 0.00 :          401ffd:       cvtpd2ps %xmm2,%xmm2
      :            return QVector2D(-vector.xp, -vector.yp, 1);
      :        }
      :
      :        inline const QVector2D operator/(const QVector2D &vector, qreal divisor)
      :        {
      :            return QVector2D(vector.xp / divisor, vector.yp / divisor, 1);
 0.02 :          402001:       unpcklps %xmm1,%xmm1
 0.51 :          402004:       cvtps2pd %xmm1,%xmm1
      :            return v1.xp != v2.xp || v1.yp != v2.yp;
      :        }
      :
      :        inline const QVector2D operator+(const QVector2D &v1, const QVector2D &v2)
      :        {
      :            return QVector2D(v1.xp + v2.xp, v1.yp + v2.yp, 1);
 0.00 :          402007:       addss  %xmm3,%xmm2
      :
      :        inline QVector2D::QVector2D(float xpos, float ypos, int) : xp(xpos), yp(ypos) {}
      :
      :        inline QVector2D::QVector2D(qreal xpos, qreal ypos) : xp(xpos), yp(ypos) {}
      :
      :        inline QVector2D::QVector2D(const QPoint& point) : xp(point.x()), yp(point.y()) {}
 0.02 :          40200b:       cvtsi2ss %r12d,%xmm3
      :            return QVector2D(-vector.xp, -vector.yp, 1);
      :        }
      :
      :        inline const QVector2D operator/(const QVector2D &vector, qreal divisor)
      :        {
      :            return QVector2D(vector.xp / divisor, vector.yp / divisor, 1);
 0.40 :          402010:       divsd  %xmm0,%xmm1
 6.87 :          402014:       unpcklpd %xmm1,%xmm1
 0.42 :          402018:       cvtpd2ps %xmm1,%xmm1
```

```
        :            return v1.xp != v2.xp || v1.yp != v2.yp;
        :        }
        :
        :        inline const QVector2D operator+(const QVector2D &v1, const QVector2D &v2)
        :        {
        :            return QVector2D(v1.xp + v2.xp, v1.yp + v2.yp, 1);
   1.81 :          40201c:       addss  %xmm2,%xmm1
        :            return QVector2D(v1.xp - v2.xp, v1.yp - v2.yp, 1);
        :        }
        :
        :        inline const QVector2D operator*(qreal factor, const QVector2D &vector)
        :        {
        :            return QVector2D(vector.xp * factor, vector.yp * factor, 1);
   1.49 :          402020:       movss  -0x90(%rbp),%xmm2
   0.00 :          402028:       movq   $0x0,-0x90(%rbp)
   0.00 :          402033:       cvtps2pd %xmm2,%xmm2
   0.49 :          402036:       mulsd  -0x88(%rbp),%xmm2
   0.00 :          40203e:       unpcklpd %xmm2,%xmm2
   0.00 :          402042:       cvtpd2ps %xmm2,%xmm2
        :            return QVector2D(-vector.xp, -vector.yp, 1);
        :        }
        :
        :        inline const QVector2D operator/(const QVector2D &vector, qreal divisor)
        :        {
        :            return QVector2D(vector.xp / divisor, vector.yp / divisor, 1);
   0.47 :          402046:       unpcklps %xmm2,%xmm2
   0.00 :          402049:       cvtps2pd %xmm2,%xmm2
   0.00 :          40204c:       divsd  %xmm0,%xmm2
   5.91 :          402050:       unpcklpd %xmm2,%xmm2
   0.36 :          402054:       cvtpd2ps %xmm2,%xmm0
        :            return QVector2D(v1.xp - v2.xp, v1.yp - v2.yp, 1);
        :        }
        :
        :        inline const QVector2D operator*(qreal factor, const QVector2D &vector)
        :        {
        :            return QVector2D(vector.xp * factor, vector.yp * factor, 1);
   1.98 :          402058:       movss  -0x3c(%rbp),%xmm2
   0.00 :          40205d:       cvtps2pd %xmm2,%xmm2
   0.04 :          402060:       mulsd  -0x78(%rbp),%xmm2
   0.44 :          402065:       unpcklpd %xmm2,%xmm2
   0.00 :          402069:       cvtpd2ps %xmm2,%xmm2
        :            return v1.xp != v2.xp || v1.yp != v2.yp;
        :        }
        :
        :        inline const QVector2D operator+(const QVector2D &v1, const QVector2D &v2)
        :        {
        :            return QVector2D(v1.xp + v2.xp, v1.yp + v2.yp, 1);
   0.04 :          40206d:       addss  %xmm3,%xmm2
   0.47 :          402071:       addss  %xmm2,%xmm0
   0.47 :          402075:       xorpd  %xmm2,%xmm2
        :            return qFuzzyCompare(v1.xp, v2.xp) && qFuzzyCompare(v1.yp, v2.yp);
        :        }
        :
        :        inline QPoint QVector2D::toPoint() const
        :        {
        :            return QPoint(qRound(xp), qRound(yp));
   0.00 :          402079:       unpcklps %xmm0,%xmm0
   0.49 :          40207c:       cvtps2pd %xmm0,%xmm0
   0.96 :          40207f:       ucomisd %xmm2,%xmm0
   0.83 :          402083:       jae    401d20 <Model::morph(int, double, double, double)+0x1a0>
   0.05 :          402089:       movapd %xmm0,%xmm2
   0.02 :          40208d:       unpcklps %xmm1,%xmm1
   0.00 :          402090:       subsd  0x8c0(%rip),%xmm2        # 402958 <_IO_stdin_used+0xa8>
   0.07 :          402098:       xorpd  %xmm3,%xmm3
```

```
 0.01 :          40209c:       cvtps2pd %xmm1,%xmm1
 0.00 :          40209f:       cvttsd2si %xmm2,%eax
 0.24 :          4020a3:       cvtsi2sd %eax,%xmm2
 0.21 :          4020a7:       subsd  %xmm2,%xmm0
 0.12 :          4020ab:       movsd  0x85d(%rip),%xmm2        # 402910 <_IO_stdin_used+0x60>
 0.00 :          4020b3:       movsd  %xmm2,-0x80(%rbp)
 0.00 :          4020b8:       addsd  %xmm2,%xmm0
 0.17 :          4020bc:       cvttsd2si %xmm0,%r12d
 0.20 :          4020c1:       add    %eax,%r12d
 0.08 :          4020c4:       ucomisd %xmm3,%xmm1
 0.00 :          4020c8:       jae    401d4a <Model::morph(int, double, double, double)+0x1ca>
 0.00 :          4020ce:       movapd %xmm1,%xmm0
 0.01 :          4020d2:       subsd  0x87e(%rip),%xmm0        # 402958 <_IO_stdin_used+0xa8>
 0.00 :          4020da:       cvttsd2si %xmm0,%eax
 0.00 :          4020de:       cvtsi2sd %eax,%xmm0
 0.01 :          4020e2:       subsd  %xmm0,%xmm1
 0.00 :          4020e6:       addsd  -0x80(%rbp),%xmm1
 0.00 :          4020eb:       cvttsd2si %xmm1,%r13d
 0.01 :          4020f0:       add    %eax,%r13d
 0.00 :          4020f3:       jmpq   401d54 <Model::morph(int, double, double, double)+0x1d4>
 0.00 :          4020f8:       nopl   0x0(%rax,%rax,1)
      :
      :                        QPoint p = X2.toPoint() - X;
      :
      :                        double dist = 0;
      :                        if(u > 0 && u < 1) dist = fabs(v);
      :                        else if(u <= 0) dist = sqrt(pow(X.x() - P.x(), 2.0) + pow(X.y() - P.y(), 2.0));
 0.00 :          402100:       cvtsi2sdl -0xa4(%rbp),%xmm1
 0.01 :          402108:       cvtsi2sdl -0xa0(%rbp),%xmm0
 0.15 :          402110:       jmpq   401db0 <Model::morph(int, double, double, double)+0x230>
 0.00 :          402115:       nopl   (%rax)
      :                        QVector2D X2 = QVector2D(P2) + u * Q2P2 + (v * pQ2P2) / Q2P2.length();
      :
      :                        QPoint p = X2.toPoint() - X;
      :
      :                        double dist = 0;
      :                        if(u > 0 && u < 1) dist = fabs(v);
 0.00 :          402118:       movsd  0x810(%rip),%xmm0        # 402930 <_IO_stdin_used+0x80>
 0.01 :          402120:       movsd  -0x88(%rbp),%xmm2
 0.03 :          402128:       andpd  %xmm0,%xmm2
 0.20 :          40212c:       movsd  %xmm2,-0x78(%rbp)
 0.00 :          402131:       jmpq   401dcf <Model::morph(int, double, double, double)+0x24f>
 0.00 :          402136:       nopw   %cs:0x0(%rax,%rax,1)
 0.00 :          402140:       movsd  0x810(%rip),%xmm4        # 402958 <_IO_stdin_used+0xa8>
      :
      :                    double ww[lines];
      :                    QPoint pp[lines];
      :
      :                    // for each line
      :                    for(int k=0; k<lines; ++k) {
 0.01 :          402148:        xor    %eax,%eax
 0.09 :          40214a:        xor    %ecx,%ecx
 0.00 :          40214c:        xor    %esi,%esi
 0.00 :          40214e:        mov    -0xe4(%rbp),%r8d
 0.00 :          402155:        xorpd  %xmm2,%xmm2
 0.12 :          402159:        mov    -0xc0(%rbp),%r9
 0.00 :          402160:        movapd %xmm4,%xmm5
 0.00 :          402164:        mov    %rdx,%r10
 0.00 :          402167:        jmp    4021a5 <Model::morph(int, double, double, double)+0x625>
 0.00 :          402169:        nopl   0x0(%rax)
 0.15 :          402170:        addsd  -0x80(%rbp),%xmm0
 0.11 :          402175:        cvttsd2si %xmm0,%edi
      :
      :       inline const QPoint operator*(float c, const QPoint &p)
```

```
        :        { return QPoint(qRound(p.xp*c), qRound(p.yp*c)); }
        :
        :        inline const QPoint operator*(double c, const QPoint &p)
        :        { return QPoint(qRound(p.xp*c), qRound(p.yp*c)); }
   0.72 :          402179:       cvtsi2sdl (%rdx),%xmm0
   0.07 :          40217d:       mulsd  %xmm1,%xmm0
   1.77 :          402181:       xorpd  %xmm3,%xmm3
   0.00 :          402185:       ucomisd %xmm3,%xmm0
   0.85 :          402189:       jb     4021f0 <Model::morph(int, double, double, double)+0x670>
   0.29 :          40218b:       addsd  -0x80(%rbp),%xmm0
   0.27 :          402190:       cvttsd2si %xmm0,%edx
   0.96 :          402194:       add    $0x1,%rax
        :
        :        inline int &QPoint::ry()
        :        { return yp; }
        :
        :        inline QPoint &QPoint::operator+=(const QPoint &p)
        :        { xp+=p.xp; yp+=p.yp; return *this; }
   0.01 :          402198:       add    %edx,%esi
   0.46 :          40219a:       add    %edi,%ecx
        :                      pp[k] = p;
        :                  }
        :
        :                  QPoint sum(0.0, 0.0);
        :                  double wsum = 0;
        :                  for(int k=0; k<lines; ++k) {
   0.01 :          40219c:       cmp    %eax,%r8d
        :                      sum  += ww[k] * pp[k];
        :                      wsum += ww[k];
   0.00 :          40219f:       addsd  %xmm1,%xmm2
        :                      pp[k] = p;
        :                  }
        :
        :                  QPoint sum(0.0, 0.0);
        :                  double wsum = 0;
        :                  for(int k=0; k<lines; ++k) {
   0.01 :          4021a3:       jle    402218 <Model::morph(int, double, double, double)+0x698>
        :                      sum  += ww[k] * pp[k];
   0.48 :          4021a5:       movslq %eax,%rdx
   0.03 :          4021a8:       movsd  (%r9,%rax,8),%xmm1
   0.00 :          4021ae:       lea    (%r10,%rdx,8),%rdx
   0.01 :          4021b2:       xorpd  %xmm3,%xmm3
        :
        :        inline const QPoint operator*(float c, const QPoint &p)
        :        { return QPoint(qRound(p.xp*c), qRound(p.yp*c)); }
        :
        :        inline const QPoint operator*(double c, const QPoint &p)
        :        { return QPoint(qRound(p.xp*c), qRound(p.yp*c)); }
   0.43 :          4021b6:       cvtsi2sdl 0x4(%rdx),%xmm0
   0.04 :          4021bb:       mulsd  %xmm1,%xmm0
   0.65 :          4021bf:       ucomisd %xmm3,%xmm0
   1.01 :          4021c3:       jae    402170 <Model::morph(int, double, double, double)+0x5f0>
   0.21 :          4021c5:       movapd %xmm0,%xmm3
   0.00 :          4021c9:       subsd  %xmm4,%xmm3
   0.26 :          4021cd:       cvttsd2si %xmm3,%edi
   0.93 :          4021d1:       cvtsi2sd %edi,%xmm3
   0.91 :          4021d5:       subsd  %xmm3,%xmm0
   0.63 :          4021d9:       addsd  -0x80(%rbp),%xmm0
   0.65 :          4021de:       cvttsd2si %xmm0,%r11d
   0.84 :          4021e3:       add    %r11d,%edi
   0.19 :          4021e6:       jmp    402179 <Model::morph(int, double, double, double)+0x5f9>
   0.00 :          4021e8:       nopl   0x0(%rax,%rax,1)
   0.25 :          4021f0:       movapd %xmm0,%xmm3
   0.00 :          4021f4:       subsd  %xmm5,%xmm3
```

```
 0.16 :          4021f8:       cvttsd2si %xmm3,%edx
 0.75 :          4021fc:       cvtsi2sd %edx,%xmm3
 0.77 :          402200:       subsd  %xmm3,%xmm0
 0.60 :          402204:       addsd  -0x80(%rbp),%xmm0
 0.54 :          402209:       cvttsd2si %xmm0,%r11d
 0.85 :          40220e:       add    %r11d,%edx
 0.20 :          402211:       jmp    402194 <Model::morph(int, double, double, double)+0x614>
 0.00 :          402213:       nopl   0x0(%rax,%rax,1)
 0.17 :          402218:       cvtsi2sd %esi,%xmm1
 0.31 :          40221c:       cvtsi2sd %ecx,%xmm0
      :        inline const QPoint operator-(const QPoint &p)
      :        { return QPoint(-p.xp, -p.yp); }
      :
      :        inline QPoint &QPoint::operator/=(qreal c)
      :        {
      :            xp = qRound(xp/c);
 0.00 :          402220:       divsd  %xmm2,%xmm1
 4.19 :          402224:       ucomisd -0x90(%rbp),%xmm1
 0.19 :          40222c:       jb     402390 <Model::morph(int, double, double, double)+0x810>
 0.07 :          402232:       addsd  -0x80(%rbp),%xmm1
 0.05 :          402237:       cvttsd2si %xmm1,%eax
      :            yp = qRound(yp/c);
 0.32 :          40223b:       divsd  %xmm2,%xmm0
 0.03 :          40223f:       ucomisd -0x90(%rbp),%xmm0
 0.08 :          402247:       jb     402368 <Model::morph(int, double, double, double)+0x7e8>
 0.03 :          40224d:       addsd  -0x80(%rbp),%xmm0
 0.00 :          402252:       cvttsd2si %xmm0,%ebx
      :
      :        inline bool operator!=(const QPoint &p1, const QPoint &p2)
      :        { return p1.xp != p2.xp || p1.yp != p2.yp; }
      :
      :        inline const QPoint operator+(const QPoint &p1, const QPoint &p2)
      :        { return QPoint(p1.xp+p2.xp, p1.yp+p2.yp); }
 0.08 :          402256:       add    -0x98(%rbp),%eax
 0.03 :          40225c:       add    -0x94(%rbp),%ebx
      :                    sum /= wsum;
      :
      :                    QPoint X2 = X + sum;
      :
      :                    double y0, x0;
      :                    x0 = ceil(X2.x());
 0.01 :          402262:       cvtsi2sd %eax,%xmm0
 0.22 :          402266:       callq  400e00 <ceil@plt>
 0.03 :          40226b:       movapd %xmm0,%xmm1
      :                    if(x0 < 0) x0 = 0;
      :                    if(x0 >= wimg) x0 = wimg-1;
 0.08 :          40226f:       mov    0x4(%r14),%eax
      :
      :                    QPoint X2 = X + sum;
      :
      :                    double y0, x0;
      :                    x0 = ceil(X2.x());
      :                    if(x0 < 0) x0 = 0;
 0.00 :          402273:       cmpnltsd -0x90(%rbp),%xmm0
 0.33 :          40227c:       movsd  -0x90(%rbp),%xmm3
 0.00 :          402284:       movapd %xmm1,%xmm2
 0.00 :          402288:       movapd %xmm0,%xmm1
 0.15 :          40228c:       andpd  %xmm0,%xmm2
      :                    if(x0 >= wimg) x0 = wimg-1;
 0.11 :          402290:       cvtsi2sd %eax,%xmm0
      :
      :                    QPoint X2 = X + sum;
      :
      :                    double y0, x0;
```

```
         :                    x0 = ceil(X2.x());
         :                    if(x0 < 0) x0 = 0;
    0.13 :          402294:       andnpd %xmm3,%xmm1
    0.00 :          402298:       orpd   %xmm2,%xmm1
         :                    if(x0 >= wimg) x0 = wimg-1;
    0.12 :          40229c:       ucomisd %xmm0,%xmm1
    0.28 :          4022a0:       jb     4022a9 <Model::morph(int, double, double, double)+0x729>
    0.00 :          4022a2:       sub    $0x1,%eax
    0.00 :          4022a5:       cvtsi2sd %eax,%xmm1
         :
         :                    y0 = ceil(X2.y());
    0.17 :          4022a9:       cvtsi2sd %ebx,%xmm0
    0.00 :          4022ad:       movsd  %xmm1,-0x140(%rbp)
    0.00 :          4022b5:       callq  400e00 <ceil@plt>
         :                    if(y0 < 0) y0 = 0;
    0.16 :          4022ba:       movsd  -0x90(%rbp),%xmm2
         :                    if(y0 >= himg) y0 = himg-1;
    0.03 :          4022c2:       mov    (%r14),%eax
    0.02 :          4022c5:       movsd  -0x140(%rbp),%xmm1
         :                    x0 = ceil(X2.x());
         :                    if(x0 < 0) x0 = 0;
         :                    if(x0 >= wimg) x0 = wimg-1;
         :
         :                    y0 = ceil(X2.y());
         :                    if(y0 < 0) y0 = 0;
    0.01 :          4022cd:       maxsd  %xmm0,%xmm2
    0.28 :          4022d1:       movapd %xmm2,%xmm0
         :                    if(y0 >= himg) y0 = himg-1;
    0.11 :          4022d5:       cvtsi2sd %eax,%xmm2
    0.27 :          4022d9:       ucomisd %xmm2,%xmm0
    0.42 :          4022dd:       jb     4022e6 <Model::morph(int, double, double, double)+0x766>
    0.00 :          4022df:       sub    $0x1,%eax
    0.00 :          4022e2:       cvtsi2sd %eax,%xmm0
         :
         :                    X2 = QPoint(x0, y0);
    0.15 :          4022e6:       cvttsd2si %xmm0,%edx
         :
         :                // Inline functions...
         :
         :                Q_GUI_EXPORT_INLINE bool QImage::valid(const QPoint &pt) const { return valid(pt.x(), pt.y()); }
         :                Q_GUI_EXPORT_INLINE int QImage::pixelIndex(const QPoint &pt) const { return pixelIndex(pt.x(), pt.y());}
         :                Q_GUI_EXPORT_INLINE QRgb QImage::pixel(const QPoint &pt) const { return pixel(pt.x(), pt.y()); }
    0.00 :            4022ea:       mov    -0x118(%rbp),%rax
    0.01 :            4022f1:       cvttsd2si %xmm1,%esi
    0.11 :            4022f5:       mov    0x18(%r14,%rax,8),%rdi
    0.02 :            4022fa:       callq  400f90 <QImage::pixel(int, int) const@plt>
         :                Q_GUI_EXPORT_INLINE void QImage::setPixel(const QPoint &pt, uint index_or_rgb) { setPixel(pt.x(), pt.y(),
index_or_rgb); }
    0.01 :            4022ff:       mov    -0x120(%rbp),%rdx
    0.00 :            402306:       mov    -0x98(%rbp),%esi
    0.00 :            40230c:       mov    %eax,%ecx
    0.08 :            40230e:       mov    0x18(%r14,%rdx,8),%rdi
    0.01 :            402313:       mov    -0x94(%rbp),%edx
    0.00 :            402319:       callq  400f20 <QImage::setPixel(int, int, unsigned int)@plt>
         :                void Model::morph(int h, double VARA, double VARB, double VARP) {
         :                    int n = 0;
         :                    int lines = listLines[0]->size();
         :
         :                    for(int i=0; i<wimg; ++i) {
         :                        for(int j=0; j<himg; ++j) {
    0.04 :            40231e:       addl   $0x1,-0x94(%rbp)
    0.15 :            402325:       mov    -0xf0(%rbp),%rsp
    0.00 :            40232c:       mov    -0x94(%rbp),%ecx
    0.10 :            402332:       cmp    %ecx,(%r14)
```

```
 0.00 :          402335:       jg     401c78 <Model::morph(int, double, double, double)+0xf8>
      :
      :        void Model::morph(int h, double VARA, double VARB, double VARP) {
      :            int n = 0;
      :            int lines = listLines[0]->size();
      :
      :            for(int i=0; i<wimg; ++i) {
 0.00 :          40233b:       addl   $0x1,-0x98(%rbp)
 0.00 :          402342:       mov    -0x98(%rbp),%eax
 0.00 :          402348:       cmp    %eax,0x4(%r14)
 0.00 :          40234c:       jg     401c31 <Model::morph(int, double, double, double)+0xb1>
      :
      :                    if(X2 == X) n++;
      :                    imgs[h+2]->setPixel(X, imgs[h]->pixel(X2));
      :                }
      :            }
      :        }
 0.00 :          402352:       lea    -0x28(%rbp),%rsp
 0.00 :          402356:       pop    %rbx
 0.00 :          402357:       pop    %r12
 0.00 :          402359:       pop    %r13
 0.00 :          40235b:       pop    %r14
 0.00 :          40235d:       pop    %r15
 0.00 :          40235f:       pop    %rbp
 0.00 :          402360:       retq
 0.00 :          402361:       nopl   0x0(%rax)
 0.03 :          402368:       movapd %xmm0,%xmm1
 0.00 :          40236c:       subsd  0x5e4(%rip),%xmm1        # 402958 <_IO_stdin_used+0xa8>
 0.01 :          402374:       cvttsd2si %xmm1,%edx
 0.05 :          402378:       cvtsi2sd %edx,%xmm1
 0.01 :          40237c:       subsd  %xmm1,%xmm0
 0.02 :          402380:       addsd  -0x80(%rbp),%xmm0
 0.04 :          402385:       cvttsd2si %xmm0,%ebx
 0.02 :          402389:       add    %edx,%ebx
 0.04 :          40238b:       jmpq   402256 <Model::morph(int, double, double, double)+0x6d6>
 0.02 :          402390:       movapd %xmm1,%xmm3
 0.00 :          402394:       subsd  0x5bc(%rip),%xmm3        # 402958 <_IO_stdin_used+0xa8>
 0.03 :          40239c:       cvttsd2si %xmm3,%eax
 0.20 :          4023a0:       cvtsi2sd %eax,%xmm3
 0.25 :          4023a4:       subsd  %xmm3,%xmm1
 0.15 :          4023a8:       addsd  -0x80(%rbp),%xmm1
 0.10 :          4023ad:       cvttsd2si %xmm1,%edx
 0.21 :          4023b1:       add    %edx,%eax
 0.07 :          4023b3:       jmpq   40223b <Model::morph(int, double, double, double)+0x6bb>
      :                        QPoint p = X2.toPoint() - X;
      :
      :                        double dist = 0;
      :                        if(u > 0 && u < 1) dist = fabs(v);
      :                        else if(u <= 0) dist = sqrt(pow(X.x() - P.x(), 2.0) + pow(X.y() - P.y(), 2.0));
      :                        else dist = sqrt(pow(X.x() - Q.x(), 2.0) + pow(X.y() - Q.y(), 2.0));
 0.00 :          4023b8:       movapd %xmm1,%xmm0
 0.00 :          4023bc:       callq  400e60 <sqrt@plt>
 0.00 :          4023c1:       jmpq   401dca <Model::morph(int, double, double, double)+0x24a>
      :                /// Safety check used only from at().
      :                void
      :                _M_range_check(size_type __n) const
      :                {
      :                  if (__n >= this->size())
      :                    __throw_out_of_range(__N("vector::_M_range_check"));
 0.00 :          4023c6:       mov    $0x402918,%edi
 0.00 :          4023cb:       callq  400f10 <std::__throw_out_of_range(char const*)@plt>
 0.00 :          4023d0:       mov    $0x402918,%edi
 0.00 :          4023d5:       callq  400f10 <std::__throw_out_of_range(char const*)@plt>
      :
```

```
         :                    double ww[lines];
         :                    QPoint pp[lines];
         :
         :                    // for each line
         :                    for(int k=0; k<lines; ++k) {
    0.00 :          4023da:       xorpd  %xmm0,%xmm0
    0.00 :          4023de:       movsd  0x52a(%rip),%xmm3        # 402910 <_IO_stdin_used+0x60>
    0.00 :          4023e6:       movsd  %xmm0,-0x90(%rbp)
    0.00 :          4023ee:       movapd %xmm0,%xmm1
         :                        ww[k] = w;
         :                        pp[k] = p;
         :                    }
         :
         :                    QPoint sum(0.0, 0.0);
         :                    double wsum = 0;
    0.00 :          4023f2:       movapd %xmm0,%xmm2
    0.00 :          4023f6:       movsd  %xmm3,-0x80(%rbp)
    0.00 :          4023fb:       jmpq   402220 <Model::morph(int, double, double, double)+0x6a0>
    0.00 :          402400:       mov    %rax,%rdi
    0.00 :          402403:       callq  400f60 <_Unwind_Resume@plt>
 Percent |       Source code & Disassembly of libQtGui.so.4.8.4
-----------------------------------------------
         :
         :
         :
         :       Disassembly of section .text:
         :
         :       0000000000826b60 <QVector2D::length() const>:
    7.23 :          826b60:       movss  (%rdi),%xmm0
    0.00 :          826b64:       movss  0x4(%rdi),%xmm1
    0.02 :          826b69:       mulss  %xmm0,%xmm0
    0.18 :          826b6d:       mulss  %xmm1,%xmm1
    7.11 :          826b71:       addss  %xmm1,%xmm0
    0.09 :          826b75:       unpcklps %xmm0,%xmm0
    0.02 :          826b78:       cvtps2pd %xmm0,%xmm1
    7.23 :          826b7b:       sqrtsd %xmm1,%xmm0
   59.58 :          826b7f:       ucomisd %xmm0,%xmm0
   11.13 :          826b83:       jp     826b87 <QVector2D::length() const+0x27>
    7.39 :          826b85:       repz retq
    0.00 :          826b87:       movapd %xmm1,%xmm0
    0.00 :          826b8b:       push   %rax
    0.00 :          826b8c:       callq  1b3ac0 <sqrt@plt>
    0.00 :          826b91:       pop    %rdx
    0.00 :          826b92:       retq
 Percent |       Source code & Disassembly of libm-2.17.so
-----------------------------------------------
         :
         :
         :
         :       Disassembly of section .text:
         :
         :       0000000000025e40 <pow>:
   13.44 :          25e40:       movapd %xmm0,%xmm2
    0.06 :          25e44:       sub    $0x28,%rsp
    0.00 :          25e48:       movapd %xmm1,%xmm3
    3.28 :          25e4c:       movsd  %xmm2,0x10(%rsp)
   11.36 :          25e52:       movsd  %xmm3,(%rsp)
    0.00 :          25e57:       callq  5610 <*ABS*+0x15ae0@plt>
    0.00 :          25e5c:       movabs $0x7fffffffffffffff,%rcx
   13.94 :          25e66:       movabs $0x7feffffffffffffe,%rdx
    0.06 :          25e70:       movq   %xmm0,%rax
    0.00 :          25e75:       and    %rcx,%rax
    0.00 :          25e78:       movsd  0x10(%rsp),%xmm2
   14.32 :          25e7e:       cmp    %rdx,%rax
```

```
  0.06 :        25e81:       movsd  (%rsp),%xmm3
  0.00 :        25e86:       ja     25e97 <pow+0x57>
 13.56 :        25e88:       xorpd  %xmm1,%xmm1
  0.00 :        25e8c:       ucomisd %xmm1,%xmm0
  0.06 :        25e90:       jnp    25ee0 <pow+0xa0>
 14.32 :        25e92:       add    $0x28,%rsp
  0.00 :        25e96:       retq
  0.00 :        25e97:       mov    0x2d711a(%rip),%rsi        # 2fcfb8 <__signbitl+0x2be288>
  0.00 :        25e9e:       cmpl   $0xffffffff,(%rsi)
  0.00 :        25ea1:       je     25e92 <pow+0x52>
  0.00 :        25ea3:       movq   %xmm2,%rsi
  0.00 :        25ea8:       movabs $0x7ff0000000000000,%rdi
  0.00 :        25eb2:       and    %rcx,%rsi
  0.00 :        25eb5:       cmp    %rdi,%rsi
  0.00 :        25eb8:       jbe    25f3e <pow+0xfe>
  0.00 :        25ebe:       ucomisd 0x48802(%rip),%xmm3        # 6e6c8 <__signbitl+0x2f998>
  0.00 :        25ec6:       jp     25e92 <pow+0x52>
  0.00 :        25ec8:       jne    25e92 <pow+0x52>
  0.00 :        25eca:       movapd %xmm3,%xmm1
  0.00 :        25ece:       mov    $0x2a,%edi
  0.00 :        25ed3:       movapd %xmm2,%xmm0
  0.00 :        25ed7:       add    $0x28,%rsp
  0.00 :        25edb:       jmpq   5a80 <*ABS*+0x1f5d0@plt+0x3f0>
 15.52 :        25ee0:       jne    25e92 <pow+0x52>
  0.00 :        25ee2:       movq   %xmm2,%rax
  0.00 :        25ee7:       and    %rcx,%rax
  0.00 :        25eea:       cmp    %rdx,%rax
  0.00 :        25eed:       ja     25e92 <pow+0x52>
  0.00 :        25eef:       movq   %xmm3,%rax
  0.00 :        25ef4:       and    %rcx,%rax
  0.00 :        25ef7:       cmp    %rdx,%rax
  0.00 :        25efa:       ja     25e92 <pow+0x52>
  0.00 :        25efc:       mov    0x2d70b5(%rip),%rax        # 2fcfb8 <__signbitl+0x2be288>
  0.00 :        25f03:       cmpl   $0xffffffff,(%rax)
  0.00 :        25f06:       je     25e92 <pow+0x52>
  0.00 :        25f08:       ucomisd %xmm1,%xmm2
  0.00 :        25f0c:       jp     25f9b <pow+0x15b>
  0.00 :        25f12:       jne    25f9b <pow+0x15b>
  0.00 :        25f18:       ucomisd %xmm1,%xmm3
  0.00 :        25f1c:       jp     25e92 <pow+0x52>
  0.00 :        25f22:       jne    25e92 <pow+0x52>
  0.00 :        25f28:       movapd %xmm3,%xmm1
  0.00 :        25f2c:       mov    $0x14,%edi
  0.00 :        25f31:       movapd %xmm2,%xmm0
  0.00 :        25f35:       add    $0x28,%rsp
  0.00 :        25f39:       jmpq   5a80 <*ABS*+0x1f5d0@plt+0x3f0>
  0.00 :        25f3e:       cmp    %rdx,%rsi
  0.00 :        25f41:       ja     25e92 <pow+0x52>
  0.00 :        25f47:       movq   %xmm3,%rsi
  0.00 :        25f4c:       and    %rcx,%rsi
  0.00 :        25f4f:       cmp    %rdx,%rsi
  0.00 :        25f52:       ja     25e92 <pow+0x52>
  0.00 :        25f58:       cmp    %rdi,%rax
  0.00 :        25f5b:       ja     25f85 <pow+0x145>
  0.00 :        25f5d:       xorpd  %xmm1,%xmm1
  0.00 :        25f61:       ucomisd %xmm1,%xmm2
  0.00 :        25f65:       jp     25f6f <pow+0x12f>
  0.00 :        25f67:       jne    25f6f <pow+0x12f>
  0.00 :        25f69:       ucomisd %xmm3,%xmm1
  0.00 :        25f6d:       ja     25fb1 <pow+0x171>
  0.00 :        25f6f:       movapd %xmm3,%xmm1
  0.00 :        25f73:       mov    $0x15,%edi
  0.00 :        25f78:       movapd %xmm2,%xmm0
  0.00 :        25f7c:       add    $0x28,%rsp
```

```
    0.00 :          25f80:       jmpq   5a80 <*ABS*+0x1f5d0@plt+0x3f0>
    0.00 :          25f85:       movapd %xmm3,%xmm1
    0.00 :          25f89:       mov    $0x18,%edi
    0.00 :          25f8e:       movapd %xmm2,%xmm0
    0.00 :          25f92:       add    $0x28,%rsp
    0.00 :          25f96:       jmpq   5a80 <*ABS*+0x1f5d0@plt+0x3f0>
    0.00 :          25f9b:       movapd %xmm3,%xmm1
    0.00 :          25f9f:       mov    $0x16,%edi
    0.00 :          25fa4:       movapd %xmm2,%xmm0
    0.00 :          25fa8:       add    $0x28,%rsp
    0.00 :          25fac:       jmpq   5a80 <*ABS*+0x1f5d0@plt+0x3f0>
    0.00 :          25fb1:       pmovmskb %xmm2,%eax
    0.00 :          25fb5:       test   $0x80,%al
    0.00 :          25fb7:       je     25fc1 <pow+0x181>
    0.00 :          25fb9:       pmovmskb %xmm0,%eax
    0.00 :          25fbd:       test   $0x80,%al
    0.00 :          25fbf:       jne    25fd7 <pow+0x197>
    0.00 :          25fc1:       movapd %xmm3,%xmm1
    0.00 :          25fc5:       mov    $0x2b,%edi
    0.00 :          25fca:       movapd %xmm2,%xmm0
    0.00 :          25fce:       add    $0x28,%rsp
    0.00 :          25fd2:       jmpq   5a80 <*ABS*+0x1f5d0@plt+0x3f0>
    0.00 :          25fd7:       movapd %xmm3,%xmm1
    0.00 :          25fdb:       mov    $0x17,%edi
    0.00 :          25fe0:       movapd %xmm2,%xmm0
    0.00 :          25fe4:       add    $0x28,%rsp
    0.00 :          25fe8:       jmpq   5a80 <*ABS*+0x1f5d0@plt+0x3f0>
 Percent |       Source code & Disassembly of libQtGui.so.4.8.4
-----------------------------------------------
         :
         :
         :
         :       Disassembly of section .text:
         :
         :       000000000027cc80 <QImage::pixel(int, int) const>:
    3.14 :          27cc80:       push   %rbp
    0.51 :          27cc81:       mov    %rdi,%rbp
    0.00 :          27cc84:       push   %rbx
    0.08 :          27cc85:       movslq %esi,%rbx
    1.61 :          27cc88:       sub    $0x18,%rsp
    0.00 :          27cc8c:       mov    0x10(%rdi),%rax
    1.44 :          27cc90:       test   %rax,%rax
    0.00 :          27cc93:       je     27cc9e <QImage::pixel(int, int) const+0x1e>
    2.12 :          27cc95:       test   %ebx,%ebx
    0.00 :          27cc97:       js     27cc9e <QImage::pixel(int, int) const+0x1e>
    0.00 :          27cc99:       cmp    0x4(%rax),%ebx
    0.00 :          27cc9c:       jl     27ccc0 <QImage::pixel(int, int) const+0x40>
    0.00 :          27cc9e:       lea    0x5c8953(%rip),%rdi        # 8455f8 <typeinfo name for QBitmap+0xce8>
    0.00 :          27cca5:       xor    %eax,%eax
    0.00 :          27cca7:       mov    %ebx,%esi
    0.00 :          27cca9:       callq  1b25f0 <qWarning(char const*, ...)@plt>
    0.00 :          27ccae:       mov    $0x3039,%eax
    0.08 :          27ccb3:       add    $0x18,%rsp
    0.00 :          27ccb7:       pop    %rbx
    0.00 :          27ccb8:       pop    %rbp
    3.14 :          27ccb9:       retq
    0.00 :          27ccba:       nopw   0x0(%rax,%rax,1)
    1.10 :          27ccc0:       test   %edx,%edx
    0.00 :          27ccc2:       js     27cc9e <QImage::pixel(int, int) const+0x1e>
    0.08 :          27ccc4:       mov    %edx,0x8(%rsp)
    2.63 :          27ccc8:       callq  27c7e0 <QImage::height() const>
    0.00 :          27cccd:       mov    0x8(%rsp),%edx
    1.27 :          27ccd1:       cmp    %eax,%edx
    0.00 :          27ccd3:       jge    27cc9e <QImage::pixel(int, int) const+0x1e>
```

```
  0.42 :         27ccd5:       mov    %edx,%esi
  0.00 :         27ccd7:       mov    %rbp,%rdi
  0.17 :         27ccda:       callq  27ca20 <QImage::scanLine(int) const>
  2.63 :         27ccdf:       mov    0x10(%rbp),%rdx
  0.34 :         27cce3:       cmpl   $0xf,0x30(%rdx)
  0.25 :         27cce7:       jbe    27ccf0 <QImage::pixel(int, int) const+0x70>
  0.34 :         27cce9:       mov    (%rax,%rbx,4),%eax
 73.39 :         27ccec:       jmp    27ccb3 <QImage::pixel(int, int) const+0x33>
  0.00 :         27ccee:       xchg   %ax,%ax
  2.37 :         27ccf0:       mov    0x30(%rdx),%esi
  0.08 :         27ccf3:       lea    0x5c80f2(%rip),%rcx        # 844dec <typeinfo name for QBitmap+0x4dc>
  0.08 :         27ccfa:       movslq (%rcx,%rsi,4),%rsi
  0.85 :         27ccfe:       add    %rsi,%rcx
  1.86 :         27cd01:       jmpq   *%rcx
  0.00 :         27cd03:       movzwl (%rax,%rbx,2),%eax
  0.00 :         27cd07:       mov    %eax,%edx
  0.00 :         27cd09:       mov    %eax,%edi
  0.00 :         27cd0b:       mov    %eax,%ecx
  0.00 :         27cd0d:       and    $0xf00,%edx
  0.00 :         27cd13:       mov    %eax,%esi
  0.00 :         27cd15:       and    $0xf000,%edi
  0.00 :         27cd1b:       mov    %edx,%eax
  0.00 :         27cd1d:       sar    $0x4,%edx
  0.00 :         27cd20:       and    $0xf,%esi
  0.00 :         27cd23:       sar    $0x8,%eax
  0.00 :         27cd26:       and    $0xf0,%ecx
  0.00 :         27cd2c:       or     %edx,%eax
  0.00 :         27cd2e:       mov    %edi,%edx
  0.00 :         27cd30:       sar    $0x8,%edi
  0.00 :         27cd33:       sar    $0xc,%edx
  0.00 :         27cd36:       shl    $0x10,%eax
  0.00 :         27cd39:       or     %edi,%edx
  0.00 :         27cd3b:       shl    $0x18,%edx
  0.00 :         27cd3e:       or     %edx,%eax
  0.00 :         27cd40:       mov    %esi,%edx
  0.00 :         27cd42:       shl    $0x4,%edx
  0.00 :         27cd45:       or     %esi,%edx
  0.00 :         27cd47:       or     %edx,%eax
  0.00 :         27cd49:       mov    %ecx,%edx
  0.00 :         27cd4b:       sar    $0x4,%edx
  0.00 :         27cd4e:       or     %ecx,%edx
  0.00 :         27cd50:       shl    $0x8,%edx
  0.00 :         27cd53:       or     %edx,%eax
  0.00 :         27cd55:       jmpq   27ccb3 <QImage::pixel(int, int) const+0x33>
  0.00 :         27cd5a:       movzwl (%rax,%rbx,2),%eax
  0.00 :         27cd5e:       mov    %eax,%edx
  0.00 :         27cd60:       mov    %eax,%esi
  0.00 :         27cd62:       mov    %eax,%ecx
  0.00 :         27cd64:       and    $0xf,%edx
  0.00 :         27cd67:       and    $0xf00,%esi
  0.00 :         27cd6d:       and    $0xf0,%ecx
  0.00 :         27cd73:       mov    %edx,%eax
  0.00 :         27cd75:       shl    $0x4,%eax
  0.00 :         27cd78:       or     %edx,%eax
  0.00 :         27cd7a:       mov    %esi,%edx
  0.00 :         27cd7c:       sar    $0x4,%esi
  0.00 :         27cd7f:       sar    $0x8,%edx
  0.00 :         27cd82:       or     %esi,%edx
  0.00 :         27cd84:       shl    $0x10,%edx
  0.00 :         27cd87:       or     %edx,%eax
  0.00 :         27cd89:       mov    %ecx,%edx
  0.00 :         27cd8b:       sar    $0x4,%edx
  0.00 :         27cd8e:       or     %ecx,%edx
  0.00 :         27cd90:       shl    $0x8,%edx
```

```
0.00 :         27cd93:       or     %edx,%eax
0.00 :         27cd95:       or     $0xff000000,%eax
0.00 :         27cd9a:       jmpq   27ccb3 <QImage::pixel(int, int) const+0x33>
0.00 :         27cd9f:       lea    (%rbx,%rbx,2),%rdx
0.00 :         27cda3:       add    %rax,%rdx
0.00 :         27cda6:       movzbl (%rdx),%eax
0.00 :         27cda9:       movzbl 0x2(%rdx),%ecx
0.00 :         27cdad:       movzbl 0x1(%rdx),%edx
0.00 :         27cdb1:       shl    $0x10,%eax
0.00 :         27cdb4:       or     %ecx,%eax
0.00 :         27cdb6:       shl    $0x8,%edx
0.00 :         27cdb9:       or     %edx,%eax
0.00 :         27cdbb:       or     $0xff000000,%eax
0.00 :         27cdc0:       jmpq   27ccb3 <QImage::pixel(int, int) const+0x33>
0.00 :         27cdc5:       lea    (%rbx,%rbx,2),%rcx
0.00 :         27cdc9:       add    %rax,%rcx
0.00 :         27cdcc:       movzbl 0x2(%rcx),%eax
0.00 :         27cdd0:       movzbl 0x1(%rcx),%edx
0.00 :         27cdd4:       shl    $0x8,%eax
0.00 :         27cdd7:       or     %eax,%edx
0.00 :         27cdd9:       mov    %edx,%esi
0.00 :         27cddb:       mov    %edx,%edi
0.00 :         27cddd:       and    $0x1f,%edx
0.00 :         27cde0:       and    $0x7c00,%esi
0.00 :         27cde6:       and    $0x3e0,%edi
0.00 :         27cdec:       mov    %esi,%eax
0.00 :         27cdee:       sar    $0x7,%esi
0.00 :         27cdf1:       sar    $0xc,%eax
0.00 :         27cdf4:       or     %esi,%eax
0.00 :         27cdf6:       mov    %edx,%esi
0.00 :         27cdf8:       shl    $0x3,%edx
0.00 :         27cdfb:       sar    $0x2,%esi
0.00 :         27cdfe:       shl    $0x10,%eax
0.00 :         27ce01:       or     %edx,%esi
0.00 :         27ce03:       movzbl (%rcx),%edx
0.00 :         27ce06:       or     %esi,%eax
0.00 :         27ce08:       shl    $0x18,%edx
0.00 :         27ce0b:       or     %edx,%eax
0.00 :         27ce0d:       mov    %edi,%edx
0.00 :         27ce0f:       sar    $0x2,%edi
0.00 :         27ce12:       sar    $0x7,%edx
0.00 :         27ce15:       or     %edi,%edx
0.00 :         27ce17:       shl    $0x8,%edx
0.00 :         27ce1a:       or     %edx,%eax
0.00 :         27ce1c:       jmpq   27ccb3 <QImage::pixel(int, int) const+0x33>
0.00 :         27ce21:       movzwl (%rax,%rbx,2),%eax
0.00 :         27ce25:       mov    %eax,%edx
0.00 :         27ce27:       mov    %eax,%esi
0.00 :         27ce29:       mov    %eax,%ecx
0.00 :         27ce2b:       and    $0x1f,%edx
0.00 :         27ce2e:       and    $0x7c00,%esi
0.00 :         27ce34:       and    $0x3e0,%ecx
0.00 :         27ce3a:       mov    %edx,%eax
0.00 :         27ce3c:       shl    $0x3,%edx
0.00 :         27ce3f:       sar    $0x2,%eax
0.00 :         27ce42:       or     %edx,%eax
0.00 :         27ce44:       mov    %esi,%edx
0.00 :         27ce46:       sar    $0x7,%esi
0.00 :         27ce49:       sar    $0xc,%edx
0.00 :         27ce4c:       or     %esi,%edx
0.00 :         27ce4e:       shl    $0x10,%edx
0.00 :         27ce51:       or     %edx,%eax
0.00 :         27ce53:       mov    %ecx,%edx
0.00 :         27ce55:       sar    $0x2,%ecx
```

```
 0.00 :        27ce58:       sar    $0x7,%edx
 0.00 :        27ce5b:       or     %ecx,%edx
 0.00 :        27ce5d:       shl    $0x8,%edx
 0.00 :        27ce60:       or     %edx,%eax
 0.00 :        27ce62:       or     $0xff000000,%eax
 0.00 :        27ce67:       jmpq   27ccb3 <QImage::pixel(int, int) const+0x33>
 0.00 :        27ce6c:       lea    (%rbx,%rbx,2),%rdx
 0.00 :        27ce70:       mov    $0x3f,%edi
 0.00 :        27ce75:       add    %rdx,%rax
 0.00 :        27ce78:       movzbl (%rax),%esi
 0.00 :        27ce7b:       movzbl 0x2(%rax),%edx
 0.00 :        27ce7f:       movzbl 0x1(%rax),%ecx
 0.00 :        27ce83:       and    %esi,%edi
 0.00 :        27ce85:       lea    0x0(,%rsi,4),%eax
 0.00 :        27ce8c:       movzbl %dl,%r8d
 0.00 :        27ce90:       sar    $0x4,%edi
 0.00 :        27ce93:       mov    %r8d,%r9d
 0.00 :        27ce96:       shl    $0x6,%r8d
 0.00 :        27ce9a:       or     %edi,%eax
 0.00 :        27ce9c:       mov    $0xfffffffc,%edi
 0.00 :        27cea1:       sar    $0x6,%r9d
 0.00 :        27cea5:       and    %edx,%edi
 0.00 :        27cea7:       movzbl %al,%eax
 0.00 :        27ceaa:       and    $0x3,%edx
 0.00 :        27cead:       or     %r9d,%edi
 0.00 :        27ceb0:       or     %r8d,%edx
 0.00 :        27ceb3:       and    $0xc0,%esi
 0.00 :        27ceb9:       shl    $0x18,%edi
 0.00 :        27cebc:       sar    $0x4,%esi
 0.00 :        27cebf:       or     %edi,%eax
 0.00 :        27cec1:       mov    $0xf0,%edi
 0.00 :        27cec6:       and    %ecx,%edi
 0.00 :        27cec8:       sar    $0x2,%edi
 0.00 :        27cecb:       or     %edi,%edx
 0.00 :        27cecd:       movzbl %dl,%edx
 0.00 :        27ced0:       shl    $0x10,%edx
 0.00 :        27ced3:       or     %edx,%eax
 0.00 :        27ced5:       mov    %ecx,%edx
 0.00 :        27ced7:       and    $0xf,%edx
 0.00 :        27ceda:       sar    $0x2,%edx
 0.00 :        27cedd:       or     %edx,%esi
 0.00 :        27cedf:       mov    %ecx,%edx
 0.00 :        27cee1:       shl    $0x4,%edx
 0.00 :        27cee4:       or     %esi,%edx
 0.00 :        27cee6:       movzbl %dl,%edx
 0.00 :        27cee9:       shl    $0x8,%edx
 0.00 :        27ceec:       or     %edx,%eax
 0.00 :        27ceee:       jmpq   27ccb3 <QImage::pixel(int, int) const+0x33>
 0.00 :        27cef3:       lea    (%rbx,%rbx,2),%rdx
 0.00 :        27cef7:       mov    $0x3f,%edi
 0.00 :        27cefc:       add    %rdx,%rax
 0.00 :        27ceff:       movzbl 0x2(%rax),%ecx
 0.00 :        27cf03:       movzbl 0x1(%rax),%edx
 0.00 :        27cf07:       movzbl (%rax),%esi
 0.00 :        27cf0a:       mov    %ecx,%eax
 0.00 :        27cf0c:       and    $0x3,%ecx
 0.00 :        27cf0f:       shl    $0x6,%eax
 0.00 :        27cf12:       and    %esi,%edi
 0.00 :        27cf14:       or     %ecx,%eax
 0.00 :        27cf16:       mov    %edx,%ecx
 0.00 :        27cf18:       sar    $0x4,%edi
 0.00 :        27cf1b:       and    $0xf0,%ecx
 0.00 :        27cf21:       sar    $0x2,%ecx
 0.00 :        27cf24:       or     %ecx,%eax
```

```
0.00 :        27cf26:       lea    0x0(,%rsi,4),%ecx
0.00 :        27cf2d:       and    $0xc0,%esi
0.00 :        27cf33:       movzbl %al,%eax
0.00 :        27cf36:       sar    $0x4,%esi
0.00 :        27cf39:       or     %edi,%ecx
0.00 :        27cf3b:       shl    $0x10,%eax
0.00 :        27cf3e:       movzbl %cl,%ecx
0.00 :        27cf41:       or     %ecx,%eax
0.00 :        27cf43:       mov    %edx,%ecx
0.00 :        27cf45:       shl    $0x4,%edx
0.00 :        27cf48:       and    $0xf,%ecx
0.00 :        27cf4b:       sar    $0x2,%ecx
0.00 :        27cf4e:       or     %esi,%ecx
0.00 :        27cf50:       or     %ecx,%edx
0.00 :        27cf52:       movzbl %dl,%edx
0.00 :        27cf55:       shl    $0x8,%edx
0.00 :        27cf58:       or     %edx,%eax
0.00 :        27cf5a:       or     $0xff000000,%eax
0.00 :        27cf5f:       jmpq   27ccb3 <QImage::pixel(int, int) const+0x33>
0.00 :        27cf64:       lea    (%rbx,%rbx,2),%rdx
0.00 :        27cf68:       add    %rdx,%rax
0.00 :        27cf6b:       movzbl 0x2(%rax),%edx
0.00 :        27cf6f:       movzbl 0x1(%rax),%ecx
0.00 :        27cf73:       shl    $0x8,%edx
0.00 :        27cf76:       or     %edx,%ecx
0.00 :        27cf78:       movzbl (%rax),%edx
0.00 :        27cf7b:       mov    %ecx,%edi
0.00 :        27cf7d:       mov    %ecx,%esi
0.00 :        27cf7f:       and    $0x1f,%ecx
0.00 :        27cf82:       mov    %ecx,%eax
0.00 :        27cf84:       shl    $0x3,%ecx
0.00 :        27cf87:       and    $0xf800,%edi
0.00 :        27cf8d:       sar    $0x2,%eax
0.00 :        27cf90:       and    $0x7e0,%esi
0.00 :        27cf96:       or     %ecx,%eax
0.00 :        27cf98:       mov    %edx,%ecx
0.00 :        27cf9a:       cmp    %edx,%eax
0.00 :        27cf9c:       cmovg  %edx,%eax
0.00 :        27cf9f:       shl    $0x18,%ecx
0.00 :        27cfa2:       or     %ecx,%eax
0.00 :        27cfa4:       mov    %edi,%ecx
0.00 :        27cfa6:       sar    $0x8,%edi
0.00 :        27cfa9:       sar    $0xd,%ecx
0.00 :        27cfac:       or     %edi,%ecx
0.00 :        27cfae:       cmp    %edx,%ecx
0.00 :        27cfb0:       cmovg  %edx,%ecx
0.00 :        27cfb3:       shl    $0x10,%ecx
0.00 :        27cfb6:       or     %ecx,%eax
0.00 :        27cfb8:       mov    %esi,%ecx
0.00 :        27cfba:       sar    $0x3,%esi
0.00 :        27cfbd:       sar    $0x9,%ecx
0.00 :        27cfc0:       or     %esi,%ecx
0.00 :        27cfc2:       cmp    %edx,%ecx
0.00 :        27cfc4:       cmovle %ecx,%edx
0.00 :        27cfc7:       shl    $0x8,%edx
0.00 :        27cfca:       or     %edx,%eax
0.00 :        27cfcc:       jmpq   27ccb3 <QImage::pixel(int, int) const+0x33>
0.00 :        27cfd1:       movzwl (%rax,%rbx,2),%eax
0.00 :        27cfd5:       mov    %eax,%edx
0.00 :        27cfd7:       mov    %eax,%esi
0.00 :        27cfd9:       mov    %eax,%ecx
0.00 :        27cfdb:       and    $0x1f,%edx
0.00 :        27cfde:       and    $0xf800,%esi
0.00 :        27cfe4:       and    $0x7e0,%ecx
```

```
    0.00 :          27cfea:       mov    %edx,%eax
    0.00 :          27cfec:       shl    $0x3,%edx
    0.00 :          27cfef:       sar    $0x2,%eax
    0.00 :          27cff2:       or     %edx,%eax
    0.00 :          27cff4:       mov    %esi,%edx
    0.00 :          27cff6:       sar    $0x8,%esi
    0.00 :          27cff9:       sar    $0xd,%edx
    0.00 :          27cffc:       or     %esi,%edx
    0.00 :          27cffe:       shl    $0x10,%edx
    0.00 :          27d001:       or     %edx,%eax
    0.00 :          27d003:       mov    %ecx,%edx
    0.00 :          27d005:       sar    $0x3,%ecx
    0.00 :          27d008:       sar    $0x9,%edx
    0.00 :          27d00b:       or     %ecx,%edx
    0.00 :          27d00d:       shl    $0x8,%edx
    0.00 :          27d010:       or     %edx,%eax
    0.00 :          27d012:       or     $0xff000000,%eax
    0.00 :          27d017:       jmpq   27ccb3 <QImage::pixel(int, int) const+0x33>
    0.00 :          27d01c:       movzbl (%rax,%rbx,1),%eax
    0.00 :          27d020:       mov    0x18(%rdx),%rdx
    0.00 :          27d024:       mov    0x10(%rdx,%rax,4),%eax
    0.00 :          27d028:       jmpq   27ccb3 <QImage::pixel(int, int) const+0x33>
    0.00 :          27d02d:       mov    %ebx,%ecx
    0.00 :          27d02f:       mov    0x18(%rdx),%rdx
    0.00 :          27d033:       sar    $0x3,%ecx
    0.00 :          27d036:       movslq %ecx,%rcx
    0.00 :          27d039:       movzbl (%rax,%rcx,1),%eax
    0.00 :          27d03d:       mov    %ebx,%ecx
    0.00 :          27d03f:       and    $0x7,%ecx
    0.00 :          27d042:       sar    %cl,%eax
    0.00 :          27d044:       and    $0x1,%eax
    0.00 :          27d047:       add    $0x4,%rax
    0.00 :          27d04b:       mov    (%rdx,%rax,4),%eax
    0.00 :          27d04e:       jmpq   27ccb3 <QImage::pixel(int, int) const+0x33>
    0.00 :          27d053:       mov    %ebx,%ecx
    0.00 :          27d055:       mov    0x18(%rdx),%rdx
    0.00 :          27d059:       sar    $0x3,%ecx
    0.00 :          27d05c:       movslq %ecx,%rcx
    0.00 :          27d05f:       movzbl (%rax,%rcx,1),%eax
    0.00 :          27d063:       mov    %ebx,%ecx
    0.00 :          27d065:       not    %ecx
    0.00 :          27d067:       jmp    27d03f <QImage::pixel(int, int) const+0x3bf>
 Percent |       Source code & Disassembly of libQtGui.so.4.8.4
-----------------------------------------------
         :
         :
         :
         :       Disassembly of section .text:
         :
         :       0000000000826dc0 <QVector2D::dotProduct(QVector2D const&, QVector2D const&)>:
   25.90 :          826dc0:       movss  (%rdi),%xmm0
    0.00 :          826dc4:       movss  0x4(%rdi),%xmm1
    0.11 :          826dc9:       mulss  (%rsi),%xmm0
   12.95 :          826dcd:       mulss  0x4(%rsi),%xmm1
   17.91 :          826dd2:       addss  %xmm1,%xmm0
    9.57 :          826dd6:       unpcklps %xmm0,%xmm0
    2.48 :          826dd9:       cvtps2pd %xmm0,%xmm0
   31.08 :          826ddc:       retq
 Percent |       Source code & Disassembly of libQtGui.so.4.8.4
-----------------------------------------------
         :
         :
         :
         :       Disassembly of section .text:
```

```
         :
         :        0000000000826ba0 <QVector2D::lengthSquared() const>:
   23.43 :          826ba0:       movss  (%rdi),%xmm0
    6.28 :          826ba4:       movss  0x4(%rdi),%xmm1
    0.00 :          826ba9:       mulss  %xmm0,%xmm0
   11.04 :          826bad:       mulss  %xmm1,%xmm1
   19.52 :          826bb1:       addss  %xmm1,%xmm0
   10.70 :          826bb5:       unpcklps %xmm0,%xmm0
    3.06 :          826bb8:       cvtps2pd %xmm0,%xmm0
   25.98 :          826bbb:       retq
 Percent |        Source code & Disassembly of libQtGui.so.4.8.4
-----------------------------------------------
         :
         :
         :
         :        Disassembly of section .text:
         :
         :        00000000002833f0 <QImage::setPixel(int, int, unsigned int)>:
    6.68 :          2833f0:       mov    %rbx,-0x20(%rsp)
    3.34 :          2833f5:       mov    %rbp,-0x18(%rsp)
    1.78 :          2833fa:       movslq %esi,%rbx
    6.68 :          2833fd:       mov    %r12,-0x10(%rsp)
    0.00 :          283402:       mov    %r13,-0x8(%rsp)
    0.00 :          283407:       sub    $0x38,%rsp
    0.45 :          28340b:       cmpq   $0x0,0x10(%rdi)
    8.24 :          283410:       mov    %rdi,%rbp
    0.00 :          283413:       je     283480 <QImage::setPixel(int, int, unsigned int)+0x90>
    0.45 :          283415:       test   %ebx,%ebx
    0.00 :          283417:       js     283480 <QImage::setPixel(int, int, unsigned int)+0x90>
    0.00 :          283419:       mov    %edx,0x8(%rsp)
    6.01 :          28341d:       mov    %ecx,%r12d
    0.00 :          283420:       callq  27c7c0 <QImage::width() const>
    0.00 :          283425:       cmp    %eax,%ebx
    3.79 :          283427:       mov    0x8(%rsp),%edx
    0.00 :          28342b:       jge    283480 <QImage::setPixel(int, int, unsigned int)+0x90>
    0.22 :          28342d:       test   %edx,%edx
    0.00 :          28342f:       js     283480 <QImage::setPixel(int, int, unsigned int)+0x90>
    4.90 :          283431:       mov    %rbp,%rdi
    2.90 :          283434:       callq  27c7e0 <QImage::height() const>
    1.11 :          283439:       mov    0x8(%rsp),%edx
    4.01 :          28343d:       cmp    %eax,%edx
    0.00 :          28343f:       jge    283480 <QImage::setPixel(int, int, unsigned int)+0x90>
    0.67 :          283441:       mov    %edx,%esi
    0.22 :          283443:       mov    %rbp,%rdi
    4.23 :          283446:       callq  2833a0 <QImage::scanLine(int)>
    1.34 :          28344b:       test   %rax,%rax
    0.22 :          28344e:       mov    %rax,%r13
    0.45 :          283451:       je     283736 <QImage::setPixel(int, int, unsigned int)+0x346>
    4.01 :          283457:       mov    0x10(%rbp),%rdx
    1.11 :          28345b:       cmpl   $0xf,0x30(%rdx)
    1.78 :          28345f:       jbe    2834b0 <QImage::setPixel(int, int, unsigned int)+0xc0>
    3.12 :          283461:       mov    0x18(%rsp),%rbx
    3.34 :          283466:       mov    0x20(%rsp),%rbp
    1.78 :          28346b:       mov    0x28(%rsp),%r12
    2.23 :          283470:       mov    0x30(%rsp),%r13
    2.23 :          283475:       add    $0x38,%rsp
    0.00 :          283479:       retq
    0.00 :          28347a:       nopw   0x0(%rax,%rax,1)
    0.00 :          283480:       mov    %ebx,%esi
    0.00 :          283482:       mov    0x20(%rsp),%rbp
    0.00 :          283487:       mov    0x18(%rsp),%rbx
    0.00 :          28348c:       mov    0x28(%rsp),%r12
    0.00 :          283491:       mov    0x30(%rsp),%r13
    0.00 :          283496:       lea    0x5c220b(%rip),%rdi        # 8456a8 <typeinfo name for QBitmap+0xd98>
```

```
 0.00 :        28349d:       xor    %eax,%eax
 0.00 :        28349f:       add    $0x38,%rsp
 0.00 :        2834a3:       jmpq   1b25f0 <qWarning(char const*, ...)@plt>
 0.00 :        2834a8:       nopl   0x0(%rax,%rax,1)
 1.11 :        2834b0:       mov    0x30(%rdx),%eax
 4.23 :        2834b3:       lea    0x5c19ee(%rip),%rcx        # 844ea8 <typeinfo name for QBitmap+0x598>
 2.23 :        2834ba:       movslq (%rcx,%rax,4),%rax
 3.34 :        2834be:       add    %rax,%rcx
 1.34 :        2834c1:       jmpq   *%rcx
 0.00 :        2834c3:       nopl   0x0(%rax,%rax,1)
 0.00 :        2834c8:       mov    %r12d,%eax
 0.00 :        2834cb:       and    $0xf0,%r12d
 0.00 :        2834d2:       and    $0xf0f0f0f0,%eax
 0.00 :        2834d7:       sar    $0x4,%r12d
 0.00 :        2834db:       mov    %eax,%edx
 0.00 :        2834dd:       shr    $0x8,%edx
 0.00 :        2834e0:       and    $0xff,%edx
 0.00 :        2834e6:       shr    $0xc,%eax
 0.00 :        2834e9:       or     %edx,%r12d
 0.00 :        2834ec:       and    $0xf00,%eax
 0.00 :        2834f1:       or     %eax,%r12d
 0.00 :        2834f4:       mov    %r12w,0x0(%r13,%rbx,2)
 0.00 :        2834fa:       jmpq   283461 <QImage::setPixel(int, int, unsigned int)+0x71>
 0.00 :        2834ff:       nop
 0.00 :        283500:       lea    (%rbx,%rbx,2),%rax
 0.00 :        283504:       mov    %r12,%rdx
 0.00 :        283507:       add    %rax,%r13
 0.00 :        28350a:       mov    %r12d,%eax
 0.00 :        28350d:       shr    $0x10,%eax
 0.00 :        283510:       mov    %r12b,0x2(%r13)
 0.00 :        283514:       mov    %al,0x0(%r13)
 0.00 :        283518:       movzbl %dh,%eax
 0.00 :        28351b:       mov    %al,0x1(%r13)
 0.00 :        28351f:       jmpq   283461 <QImage::setPixel(int, int, unsigned int)+0x71>
 0.00 :        283524:       nopl   0x0(%rax)
 0.00 :        283528:       lea    (%rbx,%rbx,2),%rax
 0.00 :        28352c:       mov    %r12,%rdx
 0.00 :        28352f:       movzbl %dh,%ecx
 0.00 :        283532:       add    %rax,%r13
 0.00 :        283535:       mov    %r12d,%eax
 0.00 :        283538:       lea    0x0(,%rcx,4),%edx
 0.00 :        28353f:       shr    $0x18,%eax
 0.00 :        283542:       sar    $0x6,%ecx
 0.00 :        283545:       mov    %al,0x0(%r13)
 0.00 :        283549:       movzbl %r12b,%eax
 0.00 :        28354d:       shr    $0x11,%r12d
 0.00 :        283551:       and    $0xfffffffe0,%edx
 0.00 :        283554:       sar    $0x3,%eax
 0.00 :        283557:       and    $0x7c,%r12d
 0.00 :        28355b:       or     %edx,%eax
 0.00 :        28355d:       or     %ecx,%r12d
 0.00 :        283560:       mov    %al,0x1(%r13)
 0.00 :        283564:       mov    %r12b,0x2(%r13)
 0.00 :        283568:       jmpq   283461 <QImage::setPixel(int, int, unsigned int)+0x71>
 0.00 :        28356d:       nopl   (%rax)
 0.00 :        283570:       mov    %r12d,%edx
 0.00 :        283573:       mov    %r12d,%eax
 0.00 :        283576:       shr    $0x9,%edx
 0.00 :        283579:       shr    $0x6,%eax
 0.00 :        28357c:       and    $0x7c00,%edx
 0.00 :        283582:       and    $0x3e0,%eax
 0.00 :        283587:       and    $0xff,%r12d
 0.00 :        28358e:       or     %edx,%eax
 0.00 :        283590:       sar    $0x3,%r12d
```

```
 0.00 :        283594:       or     %r12d,%eax
 0.00 :        283597:       mov    %ax,0x0(%r13,%rbx,2)
 0.00 :        28359d:       jmpq   283461 <QImage::setPixel(int, int, unsigned int)+0x71>
 0.00 :        2835a2:       nopw   0x0(%rax,%rax,1)
 0.00 :        2835a8:       lea    (%rbx,%rbx,2),%rax
 0.00 :        2835ac:       mov    %r12d,%edx
 0.00 :        2835af:       shr    $0x6,%edx
 0.00 :        2835b2:       add    %rax,%r13
 0.00 :        2835b5:       mov    %r12d,%eax
 0.00 :        2835b8:       and    $0x3f000,%edx
 0.00 :        2835be:       shr    $0x1a,%eax
 0.00 :        2835c1:       shl    $0x12,%eax
 0.00 :        2835c4:       or     %edx,%eax
 0.00 :        2835c6:       movzbl %r12b,%edx
 0.00 :        2835ca:       shr    $0x4,%r12d
 0.00 :        2835ce:       sar    $0x2,%edx
 0.00 :        2835d1:       and    $0xfc0,%r12d
 0.00 :        2835d8:       or     %edx,%eax
 0.00 :        2835da:       or     %r12d,%eax
 0.00 :        2835dd:       movzbl %ah,%edx
 0.00 :        2835e0:       mov    %al,0x0(%r13)
 0.00 :        2835e4:       shr    $0x10,%eax
 0.00 :        2835e7:       mov    %dl,0x1(%r13)
 0.00 :        2835eb:       mov    %al,0x2(%r13)
 0.00 :        2835ef:       jmpq   283461 <QImage::setPixel(int, int, unsigned int)+0x71>
 0.00 :        2835f4:       nopl   0x0(%rax)
 0.00 :        2835f8:       lea    (%rbx,%rbx,2),%rax
 0.00 :        2835fc:       mov    %r12d,%edx
 0.00 :        2835ff:       shr    $0x6,%edx
 0.00 :        283602:       add    %rax,%r13
 0.00 :        283605:       mov    %r12d,%eax
 0.00 :        283608:       and    $0x3f000,%edx
 0.00 :        28360e:       shr    $0x4,%eax
 0.00 :        283611:       and    $0xff,%r12d
 0.00 :        283618:       and    $0xfc0,%eax
 0.00 :        28361d:       sar    $0x2,%r12d
 0.00 :        283621:       or     %edx,%eax
 0.00 :        283623:       jmp    2835da <QImage::setPixel(int, int, unsigned int)+0x1ea>
 0.00 :        283625:       nopl   (%rax)
 0.00 :        283628:       lea    (%rbx,%rbx,2),%rax
 0.00 :        28362c:       add    %rax,%r13
 0.00 :        28362f:       mov    %r12,%rax
 0.00 :        283632:       movzbl %ah,%ecx
 0.00 :        283635:       mov    %r12d,%eax
 0.00 :        283638:       shr    $0x18,%eax
 0.00 :        28363b:       lea    0x0(,%rcx,8),%edx
 0.00 :        283642:       sar    $0x5,%ecx
 0.00 :        283645:       mov    %al,0x0(%r13)
 0.00 :        283649:       movzbl %r12b,%eax
 0.00 :        28364d:       shr    $0x10,%r12d
 0.00 :        283651:       and    $0xfffffe0,%edx
 0.00 :        283654:       sar    $0x3,%eax
 0.00 :        283657:       and    $0xf8,%r12d
 0.00 :        28365e:       or     %edx,%eax
 0.00 :        283660:       or     %ecx,%r12d
 0.00 :        283663:       mov    %al,0x1(%r13)
 0.00 :        283667:       mov    %r12b,0x2(%r13)
 0.00 :        28366b:       jmpq   283461 <QImage::setPixel(int, int, unsigned int)+0x71>
 0.00 :        283670:       mov    %r12d,%edx
 0.00 :        283673:       mov    %r12d,%eax
 0.00 :        283676:       shr    $0x8,%edx
 0.00 :        283679:       shr    $0x5,%eax
 0.00 :        28367c:       and    $0xf800,%edx
 0.00 :        283682:       and    $0x7e0,%eax
```

```
 0.00 :        283687:       jmpq   283587 <QImage::setPixel(int, int, unsigned int)+0x197>
 0.00 :        28368c:       nopl   0x0(%rax)
 0.00 :        283690:       mov    %r12d,0x0(%r13,%rbx,4)
 0.00 :        283695:       jmpq   283461 <QImage::setPixel(int, int, unsigned int)+0x71>
 0.00 :        28369a:       nopw   0x0(%rax,%rax,1)
 2.90 :        2836a0:       or     $0xff000000,%r12d
 0.89 :        2836a7:       mov    %r12d,0x0(%r13,%rbx,4)
 6.68 :        2836ac:       jmpq   283461 <QImage::setPixel(int, int, unsigned int)+0x71>
 0.00 :        2836b1:       nopl   0x0(%rax)
 0.00 :        2836b8:       mov    0x18(%rdx),%rax
 0.00 :        2836bc:       cmp    0x8(%rax),%r12d
 0.00 :        2836c0:       jae    2836da <QImage::setPixel(int, int, unsigned int)+0x2ea>
 0.00 :        2836c2:       mov    %r12b,0x0(%r13,%rbx,1)
 0.00 :        2836c7:       jmpq   283461 <QImage::setPixel(int, int, unsigned int)+0x71>
 0.00 :        2836cc:       nopl   0x0(%rax)
 0.00 :        2836d0:       cmp    $0x1,%r12d
 0.00 :        2836d4:       jbe    283760 <QImage::setPixel(int, int, unsigned int)+0x370>
 0.00 :        2836da:       mov    %r12d,%esi
 0.00 :        2836dd:       mov    0x18(%rsp),%rbx
 0.00 :        2836e2:       mov    0x20(%rsp),%rbp
 0.00 :        2836e7:       mov    0x28(%rsp),%r12
 0.00 :        2836ec:       mov    0x30(%rsp),%r13
 0.00 :        2836f1:       lea    0x5c1f88(%rip),%rdi        # 845680 <typeinfo name for QBitmap+0xd70>
 0.00 :        2836f8:       xor    %eax,%eax
 0.00 :        2836fa:       add    $0x38,%rsp
 0.00 :        2836fe:       jmpq   1b25f0 <qWarning(char const*, ...)@plt>
 0.00 :        283703:       nopl   0x0(%rax,%rax,1)
 0.00 :        283708:       mov    %r12d,%eax
 0.00 :        28370b:       and    $0xf0,%r12d
 0.00 :        283712:       and    $0xf0f0f0f0,%eax
 0.00 :        283717:       sar    $0x4,%r12d
 0.00 :        28371b:       mov    %eax,%edx
 0.00 :        28371d:       shr    $0x8,%edx
 0.00 :        283720:       and    $0xff,%edx
 0.00 :        283726:       or     %edx,%r12d
 0.00 :        283729:       mov    %eax,%edx
 0.00 :        28372b:       shr    $0x18,%edx
 0.00 :        28372e:       shl    $0x8,%edx
 0.00 :        283731:       jmpq   2834e6 <QImage::setPixel(int, int, unsigned int)+0xf6>
 0.00 :        283736:       mov    0x18(%rsp),%rbx
 0.00 :        28373b:       mov    0x20(%rsp),%rbp
 0.00 :        283740:       lea    0x5c2153(%rip),%rdi        # 84589a <typeinfo name for QBitmap+0xf8a>
 0.00 :        283747:       mov    0x28(%rsp),%r12
 0.00 :        28374c:       mov    0x30(%rsp),%r13
 0.00 :        283751:       xor    %eax,%eax
 0.00 :        283753:       add    $0x38,%rsp
 0.00 :        283757:       jmpq   1b25f0 <qWarning(char const*, ...)@plt>
 0.00 :        28375c:       nopl   0x0(%rax)
 0.00 :        283760:       mov    %rbp,%rdi
 0.00 :        283763:       callq  27cac0 <QImage::format() const>
 0.00 :        283768:       mov    %ebx,%edx
 0.00 :        28376a:       mov    %ebx,%ecx
 0.00 :        28376c:       sar    $0x3,%edx
 0.00 :        28376f:       cmp    $0x2,%eax
 0.00 :        283772:       movslq %edx,%rdx
 0.00 :        283775:       je     28379e <QImage::setPixel(int, int, unsigned int)+0x3ae>
 0.00 :        283777:       not    %ecx
 0.00 :        283779:       mov    $0x1,%eax
 0.00 :        28377e:       and    $0x7,%ecx
 0.00 :        283781:       shl    %cl,%eax
 0.00 :        283783:       test   %r12d,%r12d
 0.00 :        283786:       jne    283794 <QImage::setPixel(int, int, unsigned int)+0x3a4>
 0.00 :        283788:       not    %eax
 0.00 :        28378a:       and    %al,0x0(%r13,%rdx,1)
```

```
    0.00 :          28378f:       jmpq   283461 <QImage::setPixel(int, int, unsigned int)+0x71>
    0.00 :          283794:       or     %al,0x0(%r13,%rdx,1)
    0.00 :          283799:       jmpq   283461 <QImage::setPixel(int, int, unsigned int)+0x71>
    0.00 :          28379e:       and    $0x7,%ecx
    0.00 :          2837a1:       test   %r12d,%r12d
    0.00 :          2837a4:       jne    2837b6 <QImage::setPixel(int, int, unsigned int)+0x3c6>
    0.00 :          2837a6:       mov    $0x1,%al
    0.00 :          2837a8:       shl    %cl,%eax
    0.00 :          2837aa:       not    %eax
    0.00 :          2837ac:       and    %al,0x0(%r13,%rdx,1)
    0.00 :          2837b1:       jmpq   283461 <QImage::setPixel(int, int, unsigned int)+0x71>
    0.00 :          2837b6:       mov    $0x1,%eax
    0.00 :          2837bb:       shl    %cl,%eax
    0.00 :          2837bd:       or     %al,0x0(%r13,%rdx,1)
    0.00 :          2837c2:       jmpq   283461 <QImage::setPixel(int, int, unsigned int)+0x71>
 Percent |       Source code & Disassembly of test_harness
-------------------------------------------------
         :
         :
         :
         :       Disassembly of section .plt:
         :
         :       0000000000400e20 <QVector2D::length() const@plt>:
  100.00 :          400e20:       jmpq   *0x202212(%rip)        # 603038 <_GLOBAL_OFFSET_TABLE_+0x38>
    0.00 :          400e26:       pushq  $0x4
    0.00 :          400e2b:       jmpq   400dd0 <_init+0x20>
 Percent |       Source code & Disassembly of libQtGui.so.4.8.4
-------------------------------------------------
         :
         :
         :
         :       Disassembly of section .text:
         :
         :       00000000002833a0 <QImage::scanLine(int)>:
    6.17 :          2833a0:       mov    %rbx,-0x10(%rsp)
   10.29 :          2833a5:       mov    %rbp,-0x8(%rsp)
    5.76 :          2833aa:       sub    $0x18,%rsp
    0.82 :          2833ae:       cmpq   $0x0,0x10(%rdi)
    0.82 :          2833b3:       mov    %rdi,%rbx
    9.47 :          2833b6:       je     2833e8 <QImage::scanLine(int)+0x48>
    4.94 :          2833b8:       mov    %esi,%ebp
    0.41 :          2833ba:       callq  282660 <QImage::detach()>
    2.47 :          2833bf:       mov    0x10(%rbx),%rdx
    4.12 :          2833c3:       test   %rdx,%rdx
    0.00 :          2833c6:       je     2833e8 <QImage::scanLine(int)+0x48>
    9.05 :          2833c8:       mov    0x34(%rdx),%eax
   14.81 :          2833cb:       imul   %ebp,%eax
    9.05 :          2833ce:       cltq
    1.65 :          2833d0:       add    0x20(%rdx),%rax
    8.23 :          2833d4:       mov    0x8(%rsp),%rbx
    3.70 :          2833d9:       mov    0x10(%rsp),%rbp
    0.00 :          2833de:       add    $0x18,%rsp
    8.23 :          2833e2:       retq
    0.00 :          2833e3:       nopl   0x0(%rax,%rax,1)
    0.00 :          2833e8:       xor    %eax,%eax
    0.00 :          2833ea:       jmp    2833d4 <QImage::scanLine(int)+0x34>
 Percent |       Source code & Disassembly of libQtGui.so.4.8.4
-------------------------------------------------
         :
         :
         :
         :       Disassembly of section .text:
         :
         :       0000000000282660 <QImage::detach()>:
```

```
  14.47 :          282660:       push   %rbx
   9.43 :          282661:       mov    %rdi,%rbx
   0.00 :          282664:       sub    $0x30,%rsp
   6.92 :          282668:       mov    0x10(%rdi),%rax
   4.40 :          28266c:       test   %rax,%rax
   0.00 :          28266f:       je     2826d5 <QImage::detach()+0x75>
   1.26 :          282671:       testb  $0x8,0x58(%rax)
   5.66 :          282675:       jne    2826e0 <QImage::detach()+0x80>
  10.06 :          282677:       mov    (%rax),%edx
   6.92 :          282679:       cmp    $0x1,%edx
   0.00 :          28267c:       je     282708 <QImage::detach()+0xa8>
   0.00 :          282682:       lea    0x10(%rsp),%rdi
   0.00 :          282687:       mov    %rbx,%rsi
   0.00 :          28268a:       mov    %rsp,%rdx
   0.00 :          28268d:       movl   $0x0,0x4(%rsp)
   0.00 :          282695:       movl   $0x0,(%rsp)
   0.00 :          28269c:       movl   $0xffffffff,0xc(%rsp)
   0.00 :          2826a4:       movl   $0xffffffff,0x8(%rsp)
   0.00 :          2826ac:       callq  2819d0 <QImage::copy(QRect const&) const>
   0.00 :          2826b1:       lea    0x10(%rsp),%rsi
   0.00 :          2826b6:       mov    %rbx,%rdi
   0.00 :          2826b9:       callq  2820b0 <QImage::operator=(QImage const&)>
   0.00 :          2826be:       lea    0x10(%rsp),%rdi
   0.00 :          2826c3:       callq  27ef00 <QImage::~QImage()>
   0.00 :          2826c8:       mov    0x10(%rbx),%rax
   0.00 :          2826cc:       test   %rax,%rax
   0.00 :          2826cf:       je     2826d5 <QImage::detach()+0x75>
   6.29 :          2826d1:       addl   $0x1,0x3c(%rax)
   1.26 :          2826d5:       add    $0x30,%rsp
   6.92 :          2826d9:       pop    %rbx
   8.81 :          2826da:       retq
   0.00 :          2826db:       nopl   0x0(%rax,%rax,1)
   0.00 :          2826e0:       mov    (%rax),%edx
   0.00 :          2826e2:       cmp    $0x1,%edx
   0.00 :          2826e5:       jne    282677 <QImage::detach()+0x17>
   0.00 :          2826e7:       callq  27e610 <QImage::cacheKey() const>
   0.00 :          2826ec:       mov    %rax,%rdi
   0.00 :          2826ef:       callq  2aaa20 <QImagePixmapCleanupHooks::executeImageHooks(long long)>
   0.00 :          2826f4:       mov    0x10(%rbx),%rax
   0.00 :          2826f8:       mov    (%rax),%edx
   0.00 :          2826fa:       cmp    $0x1,%edx
   0.00 :          2826fd:       jne    282682 <QImage::detach()+0x22>
   0.00 :          282703:       nopl   0x0(%rax,%rax,1)
   1.89 :          282708:       testb  $0x2,0x58(%rax)
   2.52 :          28270c:       jne    282682 <QImage::detach()+0x22>
  13.21 :          282712:       jmp    2826d1 <QImage::detach()+0x71>
   0.00 :          282714:       lea    0x10(%rsp),%rdi
   0.00 :          282719:       mov    %rax,%rbx
   0.00 :          28271c:       callq  27ef00 <QImage::~QImage()>
   0.00 :          282721:       mov    %rbx,%rdi
   0.00 :          282724:       callq  1b4800 <_Unwind_Resume@plt>
 Percent |       Source code & Disassembly of test_harness
-------------------------------------------------
         :
         :
         :
         :       Disassembly of section .plt:
         :
         :       0000000000400e90 <pow@plt>:
  100.00 :         400e90:       jmpq   *0x2021da(%rip)        # 603070 <_GLOBAL_OFFSET_TABLE_+0x70>
   0.00 :         400e96:       pushq  $0xb
   0.00 :         400e9b:       jmpq   400dd0 <_init+0x20>
 Percent |       Source code & Disassembly of libz.so.1.2.7
-------------------------------------------------
```

```
        :
        :
        :
        :       Disassembly of section .text:
        :
        :       00000000000024d0 <adler32>:
   1.46 :         24d0:       push   %r15
   0.00 :         24d2:       mov    %rdi,%rax
   0.00 :         24d5:       movzwl %di,%r15d
   0.00 :         24d9:       shr    $0x10,%rax
   0.00 :         24dd:       push   %r14
   0.00 :         24df:       and    $0xffff,%eax
   0.00 :         24e4:       cmp    $0x1,%edx
   0.00 :         24e7:       push   %r13
   0.00 :         24e9:       mov    %rsi,%r13
   0.00 :         24ec:       push   %r12
   0.00 :         24ee:       push   %rbp
   0.00 :         24ef:       push   %rbx
   0.00 :         24f0:       mov    %edx,-0xc(%rsp)
   0.00 :         24f4:       mov    %rax,-0x20(%rsp)
   0.00 :         24f9:       je     274b <adler32+0x27b>
   0.00 :         24ff:       test   %rsi,%rsi
   0.00 :         2502:       je     293f <adler32+0x46f>
   0.00 :         2508:       cmpl   $0xf,-0xc(%rsp)
   0.00 :         250d:       ja     25af <adler32+0xdf>
   0.00 :         2513:       mov    -0xc(%rsp),%eax
   0.00 :         2517:       mov    -0xc(%rsp),%edx
   0.00 :         251b:       sub    $0x1,%eax
   0.00 :         251e:       test   %edx,%edx
   0.00 :         2520:       je     2545 <adler32+0x75>
   0.00 :         2522:       lea    0x1(%rsi,%rax,1),%rdx
   0.00 :         2527:       mov    -0x20(%rsp),%rax
   0.00 :         252c:       movzbl 0x0(%r13),%ecx
   0.00 :         2531:       add    $0x1,%r13
   0.00 :         2535:       add    %rcx,%r15
   0.00 :         2538:       add    %r15,%rax
   0.00 :         253b:       cmp    %rdx,%r13
   0.00 :         253e:       jne    252c <adler32+0x5c>
   0.00 :         2540:       mov    %rax,-0x20(%rsp)
   0.00 :         2545:       mov    -0x20(%rsp),%rax
   0.00 :         254a:       lea    -0xfff1(%r15),%rcx
   0.00 :         2551:       cmp    $0xfff1,%r15
   0.00 :         2558:       movabs $0xf00e10d2fc5cd,%rdx
   0.00 :         2562:       cmovb  %r15,%rcx
   0.00 :         2566:       mul    %rdx
   0.00 :         2569:       mov    -0x20(%rsp),%rax
   0.00 :         256e:       sub    %rdx,%rax
   0.00 :         2571:       shr    %rax
   0.00 :         2574:       add    %rdx,%rax
   0.00 :         2577:       shr    $0xf,%rax
   0.00 :         257b:       mov    %rax,%rsi
   0.00 :         257e:       mov    %rax,%rdx
   0.00 :         2581:       shl    $0x4,%rsi
   0.00 :         2585:       shl    $0x10,%rdx
   0.00 :         2589:       sub    %rsi,%rdx
   0.00 :         258c:       add    %rdx,%rax
   0.00 :         258f:       mov    -0x20(%rsp),%rdx
   0.00 :         2594:       sub    %rax,%rdx
   0.00 :         2597:       mov    %rdx,%rax
   0.00 :         259a:       shl    $0x10,%rax
   0.00 :         259e:       or     %rax,%rcx
   0.00 :         25a1:       pop    %rbx
   0.00 :         25a2:       pop    %rbp
   0.00 :         25a3:       pop    %r12
```

```
 0.00 :        25a5:       pop    %r13
 0.00 :        25a7:       pop    %r14
 0.00 :        25a9:       mov    %rcx,%rax
 0.00 :        25ac:       pop    %r15
 0.00 :        25ae:       retq
 0.73 :        25af:       cmpl   $0x15af,-0xc(%rsp)
 0.00 :        25b7:       jbe    279c <adler32+0x2cc>
 0.00 :        25bd:       subl   $0x15b0,-0xc(%rsp)
 0.73 :        25c5:       lea    0x15b0(%r13),%rax
 0.00 :        25cc:       mov    %r13,%r11
 0.00 :        25cf:       mov    %rax,-0x18(%rsp)
 0.00 :        25d4:       nopl   0x0(%rax)
 0.00 :        25d8:       movzbl (%r11),%r13d
 0.73 :        25dc:       movzbl 0x1(%r11),%r14d
 0.00 :        25e1:       movzbl 0x2(%r11),%r12d
 2.19 :        25e6:       movzbl 0x3(%r11),%ebp
 0.73 :        25eb:       movzbl 0x4(%r11),%ebx
 0.00 :        25f0:       movzbl 0x5(%r11),%r10d
 0.73 :        25f5:       movzbl 0x6(%r11),%r9d
 1.46 :        25fa:       movzbl 0x7(%r11),%r8d
 0.00 :        25ff:       add    %r15,%r13
 0.00 :        2602:       movzbl 0x8(%r11),%eax
 0.00 :        2607:       movzbl 0x9(%r11),%edi
 5.11 :        260c:       add    %r13,%r14
 0.00 :        260f:       movzbl 0xb(%r11),%esi
 0.00 :        2614:       movzbl 0xc(%r11),%ecx
 0.00 :        2619:       add    %r14,%r12
 2.19 :        261c:       add    %r14,%r13
 0.73 :        261f:       movzbl 0xd(%r11),%edx
 0.00 :        2624:       add    %r12,%rbp
 0.00 :        2627:       add    %r12,%r13
 4.38 :        262a:       movzbl 0xf(%r11),%r15d
 0.73 :        262f:       add    %rbp,%rbx
 0.00 :        2632:       add    %rbp,%r13
 0.73 :        2635:       add    %rbx,%r10
 2.19 :        2638:       add    %rbx,%r13
 0.73 :        263b:       add    %r10,%r9
 2.19 :        263e:       add    %r13,%r10
 0.00 :        2641:       add    %r9,%r8
 2.19 :        2644:       add    %r10,%r9
 0.00 :        2647:       add    %r8,%rax
 0.00 :        264a:       add    %r9,%r8
 0.00 :        264d:       add    %rax,%rdi
 2.19 :        2650:       mov    %rax,-0x30(%rsp)
 0.00 :        2655:       add    -0x30(%rsp),%r8
12.41 :        265a:       mov    %rdi,-0x28(%rsp)
 0.00 :        265f:       movzbl 0xa(%r11),%edi
 0.00 :        2664:       add    -0x28(%rsp),%rdi
 5.84 :        2669:       add    -0x28(%rsp),%r8
 0.00 :        266e:       movzbl 0xe(%r11),%eax
 0.00 :        2673:       add    $0x10,%r11
 0.00 :        2677:       add    %rdi,%rsi
 0.73 :        267a:       add    %r8,%rdi
 0.00 :        267d:       add    %rsi,%rcx
 4.38 :        2680:       add    %rdi,%rsi
 1.46 :        2683:       add    %rcx,%rdx
 0.73 :        2686:       add    %rsi,%rcx
 1.46 :        2689:       add    %rdx,%rax
 1.46 :        268c:       add    %rcx,%rdx
 1.46 :        268f:       add    %rax,%r15
 2.19 :        2692:       add    %rdx,%rax
 2.19 :        2695:       add    %r15,%rax
 2.19 :        2698:       add    %rax,-0x20(%rsp)
 6.57 :        269d:       cmp    %r11,-0x18(%rsp)
```

```
 0.00 :        26a2:       jne    25d8 <adler32+0x108>
 0.00 :        26a8:       movabs $0xf00e10d2fc5cd,%rax
 0.00 :        26b2:       mov    -0x18(%rsp),%r13
 0.00 :        26b7:       mul    %r15
 0.00 :        26ba:       mov    %r15,%rax
 0.00 :        26bd:       sub    %rdx,%rax
 0.00 :        26c0:       shr    %rax
 0.00 :        26c3:       add    %rdx,%rax
 0.00 :        26c6:       shr    $0xf,%rax
 0.00 :        26ca:       mov    %rax,%rcx
 0.00 :        26cd:       mov    %rax,%rdx
 0.00 :        26d0:       shl    $0x4,%rcx
 0.00 :        26d4:       shl    $0x10,%rdx
 0.00 :        26d8:       sub    %rcx,%rdx
 0.00 :        26db:       add    %rdx,%rax
 0.00 :        26de:       sub    %rax,%r15
 0.00 :        26e1:       movabs $0xf00e10d2fc5cd,%rax
 0.00 :        26eb:       mulq   -0x20(%rsp)
 0.00 :        26f0:       mov    -0x20(%rsp),%rax
 0.00 :        26f5:       sub    %rdx,%rax
 0.00 :        26f8:       shr    %rax
 0.00 :        26fb:       add    %rdx,%rax
 0.00 :        26fe:       shr    $0xf,%rax
 0.00 :        2702:       mov    %rax,%rcx
 0.00 :        2705:       mov    %rax,%rdx
 0.00 :        2708:       shl    $0x4,%rcx
 0.00 :        270c:       shl    $0x10,%rdx
 0.00 :        2710:       sub    %rcx,%rdx
 0.00 :        2713:       add    %rdx,%rax
 0.00 :        2716:       sub    %rax,-0x20(%rsp)
 0.00 :        271b:       cmpl   $0x15af,-0xc(%rsp)
 0.00 :        2723:       ja     25bd <adler32+0xed>
 0.00 :        2729:       mov    -0xc(%rsp),%eax
 0.00 :        272d:       test   %eax,%eax
 0.00 :        272f:       jne    2791 <adler32+0x2c1>
 0.00 :        2731:       mov    -0x20(%rsp),%rcx
 0.00 :        2736:       shl    $0x10,%rcx
 0.00 :        273a:       or     %r15,%rcx
 0.00 :        273d:       pop    %rbx
 0.00 :        273e:       pop    %rbp
 0.00 :        273f:       pop    %r12
 0.00 :        2741:       pop    %r13
 0.00 :        2743:       pop    %r14
 0.00 :        2745:       mov    %rcx,%rax
 0.00 :        2748:       pop    %r15
 0.00 :        274a:       retq
 0.00 :        274b:       movzbl (%rsi),%eax
 0.00 :        274e:       add    %rax,%r15
 0.00 :        2751:       lea    -0xfff1(%r15),%rax
 0.00 :        2758:       cmp    $0xfff1,%r15
 0.00 :        275f:       cmovae %rax,%r15
 0.00 :        2763:       mov    -0x20(%rsp),%rax
 0.00 :        2768:       pop    %rbx
 0.00 :        2769:       pop    %rbp
 0.00 :        276a:       add    %r15,%rax
 0.00 :        276d:       lea    -0xfff1(%rax),%rcx
 0.00 :        2774:       cmp    $0xfff1,%rax
 0.00 :        277a:       pop    %r12
 0.00 :        277c:       cmovb  %rax,%rcx
 0.00 :        2780:       pop    %r13
 0.00 :        2782:       shl    $0x10,%rcx
 0.00 :        2786:       or     %r15,%rcx
 0.00 :        2789:       pop    %r14
 0.00 :        278b:       mov    %rcx,%rax
```

```
 0.00 :        278e:       pop    %r15
 0.00 :        2790:       retq
 0.00 :        2791:       cmpl   $0xf,-0xc(%rsp)
 0.00 :        2796:       jbe    2949 <adler32+0x479>
 0.00 :        279c:       mov    -0xc(%rsp),%edx
 0.73 :        27a0:       mov    %r13,-0x8(%rsp)
 0.00 :        27a5:       mov    %edx,-0x30(%rsp)
 0.00 :        27a9:       mov    %r13,%rdx
 0.73 :        27ac:       movzbl (%rdx),%r13d
 0.73 :        27b0:       movzbl 0x1(%rdx),%r14d
 0.00 :        27b5:       movzbl 0x2(%rdx),%r12d
 0.00 :        27ba:       movzbl 0x3(%rdx),%ebp
 0.00 :        27be:       movzbl 0x4(%rdx),%ebx
 1.46 :        27c2:       movzbl 0x5(%rdx),%r11d
 0.00 :        27c7:       movzbl 0x6(%rdx),%r10d
 0.00 :        27cc:       movzbl 0x7(%rdx),%r9d
 0.00 :        27d1:       add    %r15,%r13
 0.00 :        27d4:       movzbl 0x8(%rdx),%r8d
 0.00 :        27d9:       movzbl 0x9(%rdx),%edi
 0.00 :        27dd:       add    %r13,%r14
 0.00 :        27e0:       movzbl 0xa(%rdx),%esi
 0.73 :        27e4:       movzbl 0xb(%rdx),%eax
 0.00 :        27e8:       add    %r14,%r12
 0.00 :        27eb:       add    %r14,%r13
 0.00 :        27ee:       movzbl 0xd(%rdx),%ecx
 0.73 :        27f2:       add    %r12,%rbp
 0.00 :        27f5:       add    %r13,%r12
 0.00 :        27f8:       movzbl 0xf(%rdx),%r15d
 0.00 :        27fd:       add    %rbp,%rbx
 1.46 :        2800:       add    %r12,%rbp
 0.00 :        2803:       subl   $0x10,-0x30(%rsp)
 0.00 :        2808:       add    %rbx,%r11
 0.00 :        280b:       add    %rbp,%rbx
 0.00 :        280e:       add    %r11,%r10
 0.00 :        2811:       add    %rbx,%r11
 0.00 :        2814:       add    %r10,%r9
 1.46 :        2817:       add    %r11,%r10
 0.00 :        281a:       add    %r9,%r8
 0.00 :        281d:       add    %r10,%r9
 0.73 :        2820:       add    %r8,%rdi
 0.73 :        2823:       add    %r9,%r8
 0.00 :        2826:       add    %rdi,%rsi
 0.00 :        2829:       add    %r8,%rdi
 0.00 :        282c:       add    %rsi,%rax
 0.73 :        282f:       add    %rdi,%rsi
 0.00 :        2832:       mov    %rax,-0x28(%rsp)
 0.73 :        2837:       movzbl 0xc(%rdx),%eax
 0.00 :        283b:       add    -0x28(%rsp),%rax
 2.92 :        2840:       add    -0x28(%rsp),%rsi
 0.00 :        2845:       add    %rax,%rcx
 2.19 :        2848:       mov    %rax,-0x18(%rsp)
 0.00 :        284d:       movzbl 0xe(%rdx),%eax
 0.00 :        2851:       add    -0x18(%rsp),%rsi
 4.38 :        2856:       add    %rcx,%rax
 0.00 :        2859:       add    %rsi,%rcx
 0.00 :        285c:       add    %rax,%r15
 0.00 :        285f:       add    %rcx,%rax
 0.73 :        2862:       add    %r15,%rax
 2.19 :        2865:       add    %rax,-0x20(%rsp)
 1.46 :        286a:       add    $0x10,%rdx
 0.00 :        286e:       cmpl   $0xf,-0x30(%rsp)
 0.00 :        2873:       ja     27ac <adler32+0x2dc>
 0.00 :        2879:       mov    -0xc(%rsp),%eax
 0.00 :        287d:       mov    -0xc(%rsp),%edx
```

```
    0.00 :          2881:       mov    -0x8(%rsp),%r13
    0.00 :          2886:       sub    $0x10,%eax
    0.00 :          2889:       and    $0xf,%edx
    0.00 :          288c:       shr    $0x4,%eax
    0.00 :          288f:       add    $0x1,%rax
    0.00 :          2893:       shl    $0x4,%rax
    0.00 :          2897:       add    %rax,%r13
    0.00 :          289a:       test   %edx,%edx
    0.00 :          289c:       lea    -0x1(%rdx),%eax
    0.00 :          289f:       je     28c4 <adler32+0x3f4>
    0.00 :          28a1:       lea    0x1(%r13,%rax,1),%rdx
    0.00 :          28a6:       mov    -0x20(%rsp),%rax
    0.00 :          28ab:       movzbl 0x0(%r13),%ecx
    0.00 :          28b0:       add    $0x1,%r13
    0.00 :          28b4:       add    %rcx,%r15
    0.00 :          28b7:       add    %r15,%rax
    0.00 :          28ba:       cmp    %rdx,%r13
    0.00 :          28bd:       jne    28ab <adler32+0x3db>
    0.00 :          28bf:       mov    %rax,-0x20(%rsp)
    0.00 :          28c4:       movabs $0xf00e10d2fc5cd,%rcx
    0.00 :          28ce:       mov    %r15,%rax
    0.00 :          28d1:       mul    %rcx
    0.00 :          28d4:       mov    %r15,%rax
    0.00 :          28d7:       sub    %rdx,%rax
    0.00 :          28da:       shr    %rax
    0.00 :          28dd:       add    %rdx,%rax
    0.00 :          28e0:       shr    $0xf,%rax
    0.00 :          28e4:       mov    %rax,%rsi
    0.00 :          28e7:       mov    %rax,%rdx
    0.00 :          28ea:       shl    $0x4,%rsi
    0.00 :          28ee:       shl    $0x10,%rdx
    0.00 :          28f2:       sub    %rsi,%rdx
    0.00 :          28f5:       add    %rdx,%rax
    0.00 :          28f8:       sub    %rax,%r15
    0.00 :          28fb:       mov    -0x20(%rsp),%rax
    0.00 :          2900:       mul    %rcx
    0.00 :          2903:       mov    -0x20(%rsp),%rax
    0.00 :          2908:       sub    %rdx,%rax
    0.00 :          290b:       shr    %rax
    0.00 :          290e:       add    %rdx,%rax
    0.00 :          2911:       shr    $0xf,%rax
    0.00 :          2915:       mov    %rax,%rcx
    0.00 :          2918:       mov    %rax,%rdx
    0.00 :          291b:       shl    $0x4,%rcx
    0.00 :          291f:       shl    $0x10,%rdx
    0.00 :          2923:       sub    %rcx,%rdx
    0.00 :          2926:       add    %rdx,%rax
    0.00 :          2929:       sub    %rax,-0x20(%rsp)
    0.00 :          292e:       mov    -0x20(%rsp),%rcx
    0.00 :          2933:       shl    $0x10,%rcx
    0.00 :          2937:       or     %r15,%rcx
    0.00 :          293a:       jmpq   273d <adler32+0x26d>
    0.00 :          293f:       mov    $0x1,%ecx
    0.00 :          2944:       jmpq   25a1 <adler32+0xd1>
    0.00 :          2949:       mov    -0xc(%rsp),%eax
    0.00 :          294d:       sub    $0x1,%eax
    0.00 :          2950:       jmpq   28a1 <adler32+0x3d1>
 Percent |       Source code & Disassembly of libz.so.1.2.7
-----------------------------------------------
         :
         :
         :
         :       Disassembly of section .text:
         :
```

```
       :        0000000000002b00 <crc32>:
  1.14 :          2b00:       test   %rsi,%rsi
  0.00 :          2b03:       push   %rbx
  0.00 :          2b04:       je     2dc0 <crc32+0x2c0>
  0.00 :          2b0a:       test   %edx,%edx
  0.00 :          2b0c:       not    %edi
  0.00 :          2b0e:       je     2b3d <crc32+0x3d>
  0.00 :          2b10:       test   $0x3,%sil
  0.00 :          2b14:       je     2b48 <crc32+0x48>
  0.00 :          2b16:       lea    0xbc63(%rip),%rcx        # e780 <gzclose_w+0x100>
  0.00 :          2b1d:       jmp    2b26 <crc32+0x26>
  0.00 :          2b1f:       nop
  0.00 :          2b20:       test   $0x3,%sil
  0.00 :          2b24:       je     2b48 <crc32+0x48>
  0.00 :          2b26:       movzbl (%rsi),%eax
  0.00 :          2b29:       add    $0x1,%rsi
  0.00 :          2b2d:       xor    %edi,%eax
  0.00 :          2b2f:       shr    $0x8,%edi
  0.00 :          2b32:       movzbl %al,%eax
  0.00 :          2b35:       xor    (%rcx,%rax,4),%edi
  0.00 :          2b38:       sub    $0x1,%edx
  0.00 :          2b3b:       jne    2b20 <crc32+0x20>
  0.00 :          2b3d:       mov    %edi,%eax
  0.00 :          2b3f:       not    %eax
  0.00 :          2b41:       pop    %rbx
  0.00 :          2b42:       retq
  0.00 :          2b43:       nopl   0x0(%rax,%rax,1)
  0.00 :          2b48:       cmp    $0x1f,%edx
  0.00 :          2b4b:       jbe    2d27 <crc32+0x227>
  0.00 :          2b51:       lea    0xbc28(%rip),%rcx        # e780 <gzclose_w+0x100>
  0.00 :          2b58:       mov    %edx,%r9d
  0.00 :          2b5b:       mov    %rsi,%r8
  0.00 :          2b5e:       xchg   %ax,%ax
  0.00 :          2b60:       xor    (%r8),%edi
  0.00 :          2b63:       mov    %rdi,%rax
  2.27 :          2b66:       mov    %edi,%r10d
  0.00 :          2b69:       movzbl %ah,%ebx
  1.14 :          2b6c:       movzbl %dil,%eax
  0.00 :          2b70:       shr    $0x18,%edi
  0.00 :          2b73:       mov    0xc00(%rcx,%rax,4),%eax
  3.41 :          2b7a:       xor    (%rcx,%rdi,4),%eax
  1.14 :          2b7d:       shr    $0x10,%r10d
  0.00 :          2b81:       xor    0x4(%r8),%eax
  1.14 :          2b85:       movzbl %r10b,%r10d
  0.00 :          2b89:       xor    0x800(%rcx,%rbx,4),%eax
  2.27 :          2b90:       xor    0x400(%rcx,%r10,4),%eax
  1.14 :          2b98:       movzbl %ah,%ebx
  2.27 :          2b9b:       movzbl %al,%r11d
  0.00 :          2b9f:       mov    %eax,%edi
  0.00 :          2ba1:       shr    $0x18,%eax
  0.00 :          2ba4:       shr    $0x10,%edi
  1.14 :          2ba7:       mov    %eax,%r10d
  0.00 :          2baa:       mov    0xc00(%rcx,%r11,4),%eax
  3.41 :          2bb2:       movzbl %dil,%edi
  0.00 :          2bb6:       xor    (%rcx,%r10,4),%eax
  5.68 :          2bba:       xor    0x8(%r8),%eax
  1.14 :          2bbe:       xor    0x800(%rcx,%rbx,4),%eax
  1.14 :          2bc5:       xor    0x400(%rcx,%rdi,4),%eax
  1.14 :          2bcc:       movzbl %ah,%ebx
  1.14 :          2bcf:       movzbl %al,%r11d
  0.00 :          2bd3:       mov    %eax,%edi
  0.00 :          2bd5:       shr    $0x18,%eax
  0.00 :          2bd8:       shr    $0x10,%edi
  0.00 :          2bdb:       mov    %eax,%r10d
```

```
 0.00 :        2bde:       mov    0xc00(%rcx,%r11,4),%eax
 3.41 :        2be6:       movzbl %dil,%edi
 0.00 :        2bea:       xor    (%rcx,%r10,4),%eax
 0.00 :        2bee:       xor    0xc(%r8),%eax
 0.00 :        2bf2:       xor    0x800(%rcx,%rbx,4),%eax
 2.27 :        2bf9:       xor    0x400(%rcx,%rdi,4),%eax
 2.27 :        2c00:       movzbl %ah,%ebx
 1.14 :        2c03:       movzbl %al,%r11d
 0.00 :        2c07:       mov    %eax,%edi
 0.00 :        2c09:       shr    $0x18,%eax
 0.00 :        2c0c:       shr    $0x10,%edi
 1.14 :        2c0f:       mov    %eax,%r10d
 0.00 :        2c12:       mov    0xc00(%rcx,%r11,4),%eax
 3.41 :        2c1a:       movzbl %dil,%edi
 0.00 :        2c1e:       xor    (%rcx,%r10,4),%eax
 2.27 :        2c22:       xor    0x10(%r8),%eax
 1.14 :        2c26:       xor    0x800(%rcx,%rbx,4),%eax
 0.00 :        2c2d:       xor    0x400(%rcx,%rdi,4),%eax
 1.14 :        2c34:       movzbl %ah,%ebx
 3.41 :        2c37:       movzbl %al,%r11d
 0.00 :        2c3b:       mov    %eax,%edi
 0.00 :        2c3d:       shr    $0x18,%eax
 1.14 :        2c40:       shr    $0x10,%edi
 1.14 :        2c43:       mov    %eax,%r10d
 0.00 :        2c46:       mov    0xc00(%rcx,%r11,4),%eax
 3.41 :        2c4e:       movzbl %dil,%edi
 0.00 :        2c52:       xor    (%rcx,%r10,4),%eax
 5.68 :        2c56:       xor    0x14(%r8),%eax
 0.00 :        2c5a:       xor    0x800(%rcx,%rbx,4),%eax
 2.27 :        2c61:       xor    0x400(%rcx,%rdi,4),%eax
 1.14 :        2c68:       movzbl %ah,%ebx
 3.41 :        2c6b:       movzbl %al,%r11d
 0.00 :        2c6f:       mov    %eax,%edi
 0.00 :        2c71:       shr    $0x18,%eax
 0.00 :        2c74:       shr    $0x10,%edi
 0.00 :        2c77:       mov    %eax,%r10d
 0.00 :        2c7a:       mov    0xc00(%rcx,%r11,4),%eax
 1.14 :        2c82:       movzbl %dil,%edi
 0.00 :        2c86:       xor    (%rcx,%r10,4),%eax
 2.27 :        2c8a:       sub    $0x20,%r9d
 0.00 :        2c8e:       xor    0x18(%r8),%eax
 1.14 :        2c92:       xor    0x800(%rcx,%rbx,4),%eax
 1.14 :        2c99:       xor    0x400(%rcx,%rdi,4),%eax
 0.00 :        2ca0:       movzbl %al,%r11d
 1.14 :        2ca4:       movzbl %ah,%ebx
 0.00 :        2ca7:       mov    %eax,%edi
 0.00 :        2ca9:       shr    $0x18,%eax
 0.00 :        2cac:       shr    $0x10,%edi
 0.00 :        2caf:       mov    %eax,%r10d
 0.00 :        2cb2:       mov    0xc00(%rcx,%r11,4),%eax
 4.55 :        2cba:       movzbl %dil,%edi
 0.00 :        2cbe:       xor    (%rcx,%r10,4),%eax
 2.27 :        2cc2:       xor    0x1c(%r8),%eax
 0.00 :        2cc6:       add    $0x20,%r8
 0.00 :        2cca:       xor    0x800(%rcx,%rbx,4),%eax
 0.00 :        2cd1:       xor    0x400(%rcx,%rdi,4),%eax
 1.14 :        2cd8:       mov    %eax,%edi
 0.00 :        2cda:       mov    %eax,%r10d
 0.00 :        2cdd:       movzbl %al,%r11d
 0.00 :        2ce1:       shr    $0x18,%edi
 2.27 :        2ce4:       shr    $0x10,%r10d
 0.00 :        2ce8:       movzbl %ah,%eax
 2.27 :        2ceb:       mov    (%rcx,%rdi,4),%edi
 3.41 :        2cee:       xor    0xc00(%rcx,%r11,4),%edi
```

```
    1.14 :          2cf6:       movzbl %r10b,%r10d
    0.00 :          2cfa:       xor    0x800(%rcx,%rax,4),%edi
    3.41 :          2d01:       xor    0x400(%rcx,%r10,4),%edi
    1.14 :          2d09:       cmp    $0x1f,%r9d
    0.00 :          2d0d:       ja     2b60 <crc32+0x60>
    0.00 :          2d13:       lea    -0x20(%rdx),%eax
    0.00 :          2d16:       and    $0x1f,%edx
    0.00 :          2d19:       shr    $0x5,%eax
    0.00 :          2d1c:       add    $0x1,%rax
    0.00 :          2d20:       shl    $0x5,%rax
    0.00 :          2d24:       add    %rax,%rsi
    0.00 :          2d27:       cmp    $0x3,%edx
    0.00 :          2d2a:       jbe    2d81 <crc32+0x281>
    0.00 :          2d2c:       lea    -0x4(%rdx),%eax
    0.00 :          2d2f:       lea    0xba4a(%rip),%rcx        # e780 <gzclose_w+0x100>
    0.00 :          2d36:       shr    $0x2,%eax
    0.00 :          2d39:       lea    0x4(%rsi,%rax,4),%r10
    0.00 :          2d3e:       xchg   %ax,%ax
    0.00 :          2d40:       mov    (%rsi),%eax
    0.00 :          2d42:       add    $0x4,%rsi
    0.00 :          2d46:       xor    %edi,%eax
    0.00 :          2d48:       mov    %eax,%edi
    0.00 :          2d4a:       mov    %eax,%r8d
    0.00 :          2d4d:       movzbl %al,%r9d
    0.00 :          2d51:       shr    $0x18,%edi
    0.00 :          2d54:       shr    $0x10,%r8d
    0.00 :          2d58:       movzbl %ah,%eax
    0.00 :          2d5b:       mov    (%rcx,%rdi,4),%edi
    1.14 :          2d5e:       xor    0xc00(%rcx,%r9,4),%edi
    0.00 :          2d66:       movzbl %r8b,%r8d
    0.00 :          2d6a:       xor    0x800(%rcx,%rax,4),%edi
    0.00 :          2d71:       xor    0x400(%rcx,%r8,4),%edi
    0.00 :          2d79:       cmp    %r10,%rsi
    0.00 :          2d7c:       jne    2d40 <crc32+0x240>
    0.00 :          2d7e:       and    $0x3,%edx
    0.00 :          2d81:       test   %edx,%edx
    0.00 :          2d83:       je     2b3d <crc32+0x3d>
    0.00 :          2d89:       lea    -0x1(%rdx),%eax
    0.00 :          2d8c:       lea    0xb9ed(%rip),%rcx        # e780 <gzclose_w+0x100>
    0.00 :          2d93:       lea    0x1(%rsi,%rax,1),%rdx
    0.00 :          2d98:       nopl   0x0(%rax,%rax,1)
    0.00 :          2da0:       movzbl (%rsi),%eax
    0.00 :          2da3:       add    $0x1,%rsi
    0.00 :          2da7:       xor    %edi,%eax
    0.00 :          2da9:       shr    $0x8,%edi
    0.00 :          2dac:       movzbl %al,%eax
    0.00 :          2daf:       xor    (%rcx,%rax,4),%edi
    0.00 :          2db2:       cmp    %rdx,%rsi
    0.00 :          2db5:       jne    2da0 <crc32+0x2a0>
    0.00 :          2db7:       mov    %edi,%eax
    0.00 :          2db9:       not    %eax
    0.00 :          2dbb:       jmpq   2b41 <crc32+0x41>
    0.00 :          2dc0:       xor    %eax,%eax
    0.00 :          2dc2:       pop    %rbx
    0.00 :          2dc3:       retq
 Percent |       Source code & Disassembly of libQtGui.so.4.8.4
-----------------------------------------------
         :
         :
         :
         :       Disassembly of section .text:
         :
         :       000000000027c7e0 <QImage::height() const>:
   12.50 :         27c7e0:       mov    0x10(%rdi),%rax
```

```
  32.50 :          27c7e4:       test   %rax,%rax
   0.00 :          27c7e7:       je     27c7f0 <QImage::height() const+0x10>
  42.50 :          27c7e9:       mov    0x8(%rax),%eax
  12.50 :          27c7ec:       retq
   0.00 :          27c7ed:       nopl   (%rax)
   0.00 :          27c7f0:       xor    %eax,%eax
   0.00 :          27c7f2:       retq
 Percent |       Source code & Disassembly of libc-2.17.so
-----------------------------------------------
         :
         :
         :
         :       Disassembly of section .text:
         :
         :       0000000000134e50 <__memcpy_ssse3_back>:
   0.00 :          134e50:       mov    %rdi,%rax
   1.41 :          134e53:       cmp    $0x90,%rdx
   0.00 :          134e5a:       jae    134e90 <__memcpy_ssse3_back+0x40>
   0.00 :          134e5c:       cmp    %dil,%sil
   0.00 :          134e5f:       jbe    134e7a <__memcpy_ssse3_back+0x2a>
   0.00 :          134e61:       add    %rdx,%rsi
   0.00 :          134e64:       add    %rdx,%rdi
   0.00 :          134e67:       lea    0x3d5c2(%rip),%r11        # 172430 <null+0xbe0>
   0.00 :          134e6e:       movslq (%r11,%rdx,4),%rdx
   0.00 :          134e72:       lea    (%r11,%rdx,1),%rdx
   0.00 :          134e76:       jmpq   *%rdx
   0.00 :          134e78:       ud2
   0.00 :          134e7a:       lea    0x3d36f(%rip),%r11        # 1721f0 <null+0x9a0>
   0.00 :          134e81:       movslq (%r11,%rdx,4),%rdx
   0.00 :          134e85:       lea    (%r11,%rdx,1),%rdx
   0.00 :          134e89:       jmpq   *%rdx
   0.00 :          134e8b:       ud2
   0.00 :          134e8d:       nopl   (%rax)
   0.00 :          134e90:       cmp    %dil,%sil
   0.00 :          134e93:       jle    134ef0 <__memcpy_ssse3_back+0xa0>
   0.00 :          134e95:       movdqu (%rsi),%xmm0
   0.00 :          134e99:       mov    %rdi,%r8
   0.00 :          134e9c:       and    $0xfffffffffffffff0,%rdi
   0.00 :          134ea0:       add    $0x10,%rdi
   0.00 :          134ea4:       mov    %rdi,%r9
   0.00 :          134ea7:       sub    %r8,%r9
   0.00 :          134eaa:       sub    %r9,%rdx
   0.00 :          134ead:       add    %r9,%rsi
   0.00 :          134eb0:       mov    %rsi,%r9
   0.00 :          134eb3:       and    $0xf,%r9
   0.00 :          134eb7:       je     134f50 <__memcpy_ssse3_back+0x100>
   0.00 :          134ebd:       mov    0x27233c(%rip),%rcx        # 3a7200 <__x86_64_data_cache_size>
   0.00 :          134ec4:       cmp    %rcx,%rdx
   0.00 :          134ec7:       jae    1366f0 <__memcpy_ssse3_back+0x18a0>
   0.00 :          134ecd:       lea    0x3d79c(%rip),%r11        # 172670 <null+0xe20>
   0.00 :          134ed4:       sub    $0x80,%rdx
   0.00 :          134edb:       movslq (%r11,%r9,4),%r9
   0.00 :          134edf:       add    %r11,%r9
   0.00 :          134ee2:       jmpq   *%r9
   0.00 :          134ee5:       ud2
   0.00 :          134ee7:       nopw   0x0(%rax,%rax,1)
   1.41 :          134ef0:       mov    0x272309(%rip),%rcx        # 3a7200 <__x86_64_data_cache_size>
   0.00 :          134ef7:       shl    %rcx
   0.00 :          134efa:       cmp    %rcx,%rdx
   0.00 :          134efd:       ja     136880 <__memcpy_ssse3_back+0x1a30>
   0.00 :          134f03:       add    %rdx,%rdi
   0.00 :          134f06:       add    %rdx,%rsi
   0.00 :          134f09:       movdqu -0x10(%rsi),%xmm0
   0.00 :          134f0e:       lea    -0x10(%rdi),%r8
```

```
 0.00 :         134f12:       mov    %rdi,%r9
 0.00 :         134f15:       and    $0xf,%r9
 0.00 :         134f19:       xor    %r9,%rdi
 0.00 :         134f1c:       sub    %r9,%rsi
 0.00 :         134f1f:       sub    %r9,%rdx
 0.00 :         134f22:       mov    %rsi,%r9
 0.00 :         134f25:       and    $0xf,%r9
 0.00 :         134f29:       je     134ff0 <__memcpy_ssse3_back+0x1a0>
 0.00 :         134f2f:       lea    0x3d77a(%rip),%r11        # 1726b0 <null+0xe60>
 0.00 :         134f36:       sub    $0x80,%rdx
 0.00 :         134f3d:       movslq (%r11,%r9,4),%r9
 0.00 :         134f41:       add    %r11,%r9
 0.00 :         134f44:       jmpq   *%r9
 0.00 :         134f47:       ud2
 0.00 :         134f49:       nopl   0x0(%rax)
 0.00 :         134f50:       mov    %rdx,%r9
 0.00 :         134f53:       shr    $0x8,%r9
 0.00 :         134f57:       add    %rdx,%r9
 0.00 :         134f5a:       cmp    0x2722a7(%rip),%r9        # 3a7208 <__x86_64_data_cache_size_half>
 0.00 :         134f61:       jae    1366f0 <__memcpy_ssse3_back+0x18a0>
 0.00 :         134f67:       sub    $0x80,%rdx
 0.00 :         134f6e:       xchg   %ax,%ax
 0.00 :         134f70:       movdqa (%rsi),%xmm1
 0.00 :         134f74:       movdqa %xmm1,(%rdi)
 0.00 :         134f78:       movaps 0x10(%rsi),%xmm2
 0.00 :         134f7c:       movaps %xmm2,0x10(%rdi)
 0.00 :         134f80:       movaps 0x20(%rsi),%xmm3
 0.00 :         134f84:       movaps %xmm3,0x20(%rdi)
 0.00 :         134f88:       movaps 0x30(%rsi),%xmm4
 0.00 :         134f8c:       movaps %xmm4,0x30(%rdi)
 0.00 :         134f90:       movaps 0x40(%rsi),%xmm1
 0.00 :         134f94:       movaps %xmm1,0x40(%rdi)
 0.00 :         134f98:       movaps 0x50(%rsi),%xmm2
 0.00 :         134f9c:       movaps %xmm2,0x50(%rdi)
 0.00 :         134fa0:       movaps 0x60(%rsi),%xmm3
 0.00 :         134fa4:       movaps %xmm3,0x60(%rdi)
 0.00 :         134fa8:       movaps 0x70(%rsi),%xmm4
 0.00 :         134fac:       movaps %xmm4,0x70(%rdi)
 0.00 :         134fb0:       sub    $0x80,%rdx
 0.00 :         134fb7:       lea    0x80(%rsi),%rsi
 1.41 :         134fbe:       lea    0x80(%rdi),%rdi
 0.00 :         134fc5:       jae    134f70 <__memcpy_ssse3_back+0x120>
 0.00 :         134fc7:       movdqu %xmm0,(%r8)
 0.00 :         134fcc:       add    $0x80,%rdx
 0.00 :         134fd3:       add    %rdx,%rsi
 0.00 :         134fd6:       add    %rdx,%rdi
 0.00 :         134fd9:       lea    0x3d450(%rip),%r11        # 172430 <null+0xbe0>
 0.00 :         134fe0:       movslq (%r11,%rdx,4),%rdx
 0.00 :         134fe4:       lea    (%r11,%rdx,1),%rdx
 0.00 :         134fe8:       jmpq   *%rdx
 0.00 :         134fea:       ud2
 0.00 :         134fec:       nopl   0x0(%rax)
 0.00 :         134ff0:       sub    $0x80,%rdx
 1.41 :         134ff7:       movaps -0x10(%rsi),%xmm1
 1.41 :         134ffb:       movaps %xmm1,-0x10(%rdi)
 0.00 :         134fff:       movaps -0x20(%rsi),%xmm2
 2.82 :         135003:       movaps %xmm2,-0x20(%rdi)
 0.00 :         135007:       movaps -0x30(%rsi),%xmm3
 0.00 :         13500b:       movaps %xmm3,-0x30(%rdi)
 0.00 :         13500f:       movaps -0x40(%rsi),%xmm4
 1.41 :         135013:       movaps %xmm4,-0x40(%rdi)
 0.00 :         135017:       movaps -0x50(%rsi),%xmm5
 0.00 :         13501b:       movaps %xmm5,-0x50(%rdi)
 1.41 :         13501f:       movaps -0x60(%rsi),%xmm5
```

```
 4.23 :         135023:       movaps %xmm5,-0x60(%rdi)
 1.41 :         135027:       movaps -0x70(%rsi),%xmm5
 0.00 :         13502b:       movaps %xmm5,-0x70(%rdi)
 0.00 :         13502f:       movaps -0x80(%rsi),%xmm5
 2.82 :         135033:       movaps %xmm5,-0x80(%rdi)
 1.41 :         135037:       sub    $0x80,%rdx
 0.00 :         13503e:       lea    -0x80(%rdi),%rdi
 0.00 :         135042:       lea    -0x80(%rsi),%rsi
 0.00 :         135046:       jae    134ff7 <__memcpy_ssse3_back+0x1a7>
 0.00 :         135048:       movdqu %xmm0,(%r8)
 0.00 :         13504d:       add    $0x80,%rdx
 0.00 :         135054:       sub    %rdx,%rdi
 0.00 :         135057:       sub    %rdx,%rsi
 0.00 :         13505a:       lea    0x3d18f(%rip),%r11        # 1721f0 <null+0x9a0>
 0.00 :         135061:       movslq (%r11,%rdx,4),%rdx
 0.00 :         135065:       lea    (%r11,%rdx,1),%rdx
 0.00 :         135069:       jmpq   *%rdx
 0.00 :         13506b:       ud2
 0.00 :         13506d:       nopl   (%rax)
 0.00 :         135070:       sub    $0x80,%rdx
 1.41 :         135077:       movaps -0x1(%rsi),%xmm1
 0.00 :         13507b:       movaps 0xf(%rsi),%xmm2
 0.00 :         13507f:       movaps 0x1f(%rsi),%xmm3
 0.00 :         135083:       movaps 0x2f(%rsi),%xmm4
 0.00 :         135087:       movaps 0x3f(%rsi),%xmm5
 0.00 :         13508b:       movaps 0x4f(%rsi),%xmm6
 0.00 :         13508f:       movaps 0x5f(%rsi),%xmm7
 0.00 :         135093:       movaps 0x6f(%rsi),%xmm8
 1.41 :         135098:       movaps 0x7f(%rsi),%xmm9
 0.00 :         13509d:       lea    0x80(%rsi),%rsi
 0.00 :         1350a4:       palignr $0x1,%xmm8,%xmm9
 0.00 :         1350ab:       movaps %xmm9,0x70(%rdi)
 0.00 :         1350b0:       palignr $0x1,%xmm7,%xmm8
 0.00 :         1350b7:       movaps %xmm8,0x60(%rdi)
 0.00 :         1350bc:       palignr $0x1,%xmm6,%xmm7
 0.00 :         1350c2:       movaps %xmm7,0x50(%rdi)
 0.00 :         1350c6:       palignr $0x1,%xmm5,%xmm6
 0.00 :         1350cc:       movaps %xmm6,0x40(%rdi)
 0.00 :         1350d0:       palignr $0x1,%xmm4,%xmm5
 0.00 :         1350d6:       movaps %xmm5,0x30(%rdi)
 0.00 :         1350da:       palignr $0x1,%xmm3,%xmm4
 0.00 :         1350e0:       movaps %xmm4,0x20(%rdi)
 0.00 :         1350e4:       palignr $0x1,%xmm2,%xmm3
 0.00 :         1350ea:       movaps %xmm3,0x10(%rdi)
 0.00 :         1350ee:       palignr $0x1,%xmm1,%xmm2
 0.00 :         1350f4:       movaps %xmm2,(%rdi)
 0.00 :         1350f7:       lea    0x80(%rdi),%rdi
 0.00 :         1350fe:       jae    135070 <__memcpy_ssse3_back+0x220>
 0.00 :         135104:       movdqu %xmm0,(%r8)
 0.00 :         135109:       add    $0x80,%rdx
 0.00 :         135110:       add    %rdx,%rdi
 0.00 :         135113:       add    %rdx,%rsi
 0.00 :         135116:       lea    0x3d313(%rip),%r11        # 172430 <null+0xbe0>
 0.00 :         13511d:       movslq (%r11,%rdx,4),%rdx
 0.00 :         135121:       lea    (%r11,%rdx,1),%rdx
 0.00 :         135125:       jmpq   *%rdx
 0.00 :         135127:       ud2
 0.00 :         135129:       nopl   0x0(%rax)
 0.00 :         135130:       movaps -0x1(%rsi),%xmm1
 0.00 :         135134:       movaps -0x11(%rsi),%xmm2
 0.00 :         135138:       palignr $0x1,%xmm2,%xmm1
 0.00 :         13513e:       movaps %xmm1,-0x10(%rdi)
 0.00 :         135142:       movaps -0x21(%rsi),%xmm3
 0.00 :         135146:       palignr $0x1,%xmm3,%xmm2
```

```
 0.00 :        13514c:       movaps %xmm2,-0x20(%rdi)
 0.00 :        135150:       movaps -0x31(%rsi),%xmm4
 0.00 :        135154:       palignr $0x1,%xmm4,%xmm3
 0.00 :        13515a:       movaps %xmm3,-0x30(%rdi)
 0.00 :        13515e:       movaps -0x41(%rsi),%xmm5
 0.00 :        135162:       palignr $0x1,%xmm5,%xmm4
 0.00 :        135168:       movaps %xmm4,-0x40(%rdi)
 0.00 :        13516c:       movaps -0x51(%rsi),%xmm6
 0.00 :        135170:       palignr $0x1,%xmm6,%xmm5
 0.00 :        135176:       movaps %xmm5,-0x50(%rdi)
 0.00 :        13517a:       movaps -0x61(%rsi),%xmm7
 0.00 :        13517e:       palignr $0x1,%xmm7,%xmm6
 0.00 :        135184:       movaps %xmm6,-0x60(%rdi)
 0.00 :        135188:       movaps -0x71(%rsi),%xmm8
 0.00 :        13518d:       palignr $0x1,%xmm8,%xmm7
 0.00 :        135194:       movaps %xmm7,-0x70(%rdi)
 0.00 :        135198:       movaps -0x81(%rsi),%xmm9
 0.00 :        1351a0:       palignr $0x1,%xmm9,%xmm8
 0.00 :        1351a7:       movaps %xmm8,-0x80(%rdi)
 0.00 :        1351ac:       sub    $0x80,%rdx
 0.00 :        1351b3:       lea    -0x80(%rdi),%rdi
 0.00 :        1351b7:       lea    -0x80(%rsi),%rsi
 0.00 :        1351bb:       jae    135130 <__memcpy_ssse3_back+0x2e0>
 0.00 :        1351c1:       movdqu %xmm0,(%r8)
 0.00 :        1351c6:       add    $0x80,%rdx
 0.00 :        1351cd:       sub    %rdx,%rdi
 0.00 :        1351d0:       sub    %rdx,%rsi
 0.00 :        1351d3:       lea    0x3d016(%rip),%r11        # 1721f0 <null+0x9a0>
 0.00 :        1351da:       movslq (%r11,%rdx,4),%rdx
 0.00 :        1351de:       lea    (%r11,%rdx,1),%rdx
 0.00 :        1351e2:       jmpq   *%rdx
 0.00 :        1351e4:       ud2
 0.00 :        1351e6:       nopw   %cs:0x0(%rax,%rax,1)
 0.00 :        1351f0:       sub    $0x80,%rdx
 0.00 :        1351f7:       movaps -0x2(%rsi),%xmm1
 0.00 :        1351fb:       movaps 0xe(%rsi),%xmm2
 0.00 :        1351ff:       movaps 0x1e(%rsi),%xmm3
 0.00 :        135203:       movaps 0x2e(%rsi),%xmm4
 0.00 :        135207:       movaps 0x3e(%rsi),%xmm5
 0.00 :        13520b:       movaps 0x4e(%rsi),%xmm6
 0.00 :        13520f:       movaps 0x5e(%rsi),%xmm7
 1.41 :        135213:       movaps 0x6e(%rsi),%xmm8
 0.00 :        135218:       movaps 0x7e(%rsi),%xmm9
 0.00 :        13521d:       lea    0x80(%rsi),%rsi
 0.00 :        135224:       palignr $0x2,%xmm8,%xmm9
 0.00 :        13522b:       movaps %xmm9,0x70(%rdi)
 0.00 :        135230:       palignr $0x2,%xmm7,%xmm8
 0.00 :        135237:       movaps %xmm8,0x60(%rdi)
 0.00 :        13523c:       palignr $0x2,%xmm6,%xmm7
 0.00 :        135242:       movaps %xmm7,0x50(%rdi)
 2.82 :        135246:       palignr $0x2,%xmm5,%xmm6
 0.00 :        13524c:       movaps %xmm6,0x40(%rdi)
 0.00 :        135250:       palignr $0x2,%xmm4,%xmm5
 0.00 :        135256:       movaps %xmm5,0x30(%rdi)
 0.00 :        13525a:       palignr $0x2,%xmm3,%xmm4
 0.00 :        135260:       movaps %xmm4,0x20(%rdi)
 0.00 :        135264:       palignr $0x2,%xmm2,%xmm3
 0.00 :        13526a:       movaps %xmm3,0x10(%rdi)
 0.00 :        13526e:       palignr $0x2,%xmm1,%xmm2
 0.00 :        135274:       movaps %xmm2,(%rdi)
 0.00 :        135277:       lea    0x80(%rdi),%rdi
 0.00 :        13527e:       jae    1351f0 <__memcpy_ssse3_back+0x3a0>
 0.00 :        135284:       movdqu %xmm0,(%r8)
 0.00 :        135289:       add    $0x80,%rdx
```

```
 0.00 :       135290:       add    %rdx,%rdi
 0.00 :       135293:       add    %rdx,%rsi
 0.00 :       135296:       lea    0x3d193(%rip),%r11        # 172430 <null+0xbe0>
 0.00 :       13529d:       movslq (%r11,%rdx,4),%rdx
 0.00 :       1352a1:       lea    (%r11,%rdx,1),%rdx
 0.00 :       1352a5:       jmpq   *%rdx
 0.00 :       1352a7:       ud2
 0.00 :       1352a9:       nopl   0x0(%rax)
 0.00 :       1352b0:       movaps -0x2(%rsi),%xmm1
 0.00 :       1352b4:       movaps -0x12(%rsi),%xmm2
 0.00 :       1352b8:       palignr $0x2,%xmm2,%xmm1
 0.00 :       1352be:       movaps %xmm1,-0x10(%rdi)
 0.00 :       1352c2:       movaps -0x22(%rsi),%xmm3
 0.00 :       1352c6:       palignr $0x2,%xmm3,%xmm2
 0.00 :       1352cc:       movaps %xmm2,-0x20(%rdi)
 0.00 :       1352d0:       movaps -0x32(%rsi),%xmm4
 0.00 :       1352d4:       palignr $0x2,%xmm4,%xmm3
 0.00 :       1352da:       movaps %xmm3,-0x30(%rdi)
 0.00 :       1352de:       movaps -0x42(%rsi),%xmm5
 0.00 :       1352e2:       palignr $0x2,%xmm5,%xmm4
 0.00 :       1352e8:       movaps %xmm4,-0x40(%rdi)
 0.00 :       1352ec:       movaps -0x52(%rsi),%xmm6
 0.00 :       1352f0:       palignr $0x2,%xmm6,%xmm5
 0.00 :       1352f6:       movaps %xmm5,-0x50(%rdi)
 0.00 :       1352fa:       movaps -0x62(%rsi),%xmm7
 0.00 :       1352fe:       palignr $0x2,%xmm7,%xmm6
 0.00 :       135304:       movaps %xmm6,-0x60(%rdi)
 0.00 :       135308:       movaps -0x72(%rsi),%xmm8
 0.00 :       13530d:       palignr $0x2,%xmm8,%xmm7
 0.00 :       135314:       movaps %xmm7,-0x70(%rdi)
 0.00 :       135318:       movaps -0x82(%rsi),%xmm9
 0.00 :       135320:       palignr $0x2,%xmm9,%xmm8
 0.00 :       135327:       movaps %xmm8,-0x80(%rdi)
 0.00 :       13532c:       sub    $0x80,%rdx
 0.00 :       135333:       lea    -0x80(%rdi),%rdi
 0.00 :       135337:       lea    -0x80(%rsi),%rsi
 0.00 :       13533b:       jae    1352b0 <__memcpy_ssse3_back+0x460>
 0.00 :       135341:       movdqu %xmm0,(%r8)
 0.00 :       135346:       add    $0x80,%rdx
 0.00 :       13534d:       sub    %rdx,%rdi
 0.00 :       135350:       sub    %rdx,%rsi
 0.00 :       135353:       lea    0x3ce96(%rip),%r11        # 1721f0 <null+0x9a0>
 0.00 :       13535a:       movslq (%r11,%rdx,4),%rdx
 0.00 :       13535e:       lea    (%r11,%rdx,1),%rdx
 0.00 :       135362:       jmpq   *%rdx
 0.00 :       135364:       ud2
 0.00 :       135366:       nopw   %cs:0x0(%rax,%rax,1)
 0.00 :       135370:       sub    $0x80,%rdx
 0.00 :       135377:       movaps -0x3(%rsi),%xmm1
 0.00 :       13537b:       movaps 0xd(%rsi),%xmm2
 0.00 :       13537f:       movaps 0x1d(%rsi),%xmm3
 0.00 :       135383:       movaps 0x2d(%rsi),%xmm4
 0.00 :       135387:       movaps 0x3d(%rsi),%xmm5
 0.00 :       13538b:       movaps 0x4d(%rsi),%xmm6
 0.00 :       13538f:       movaps 0x5d(%rsi),%xmm7
 0.00 :       135393:       movaps 0x6d(%rsi),%xmm8
 0.00 :       135398:       movaps 0x7d(%rsi),%xmm9
 0.00 :       13539d:       lea    0x80(%rsi),%rsi
 0.00 :       1353a4:       palignr $0x3,%xmm8,%xmm9
 0.00 :       1353ab:       movaps %xmm9,0x70(%rdi)
 0.00 :       1353b0:       palignr $0x3,%xmm7,%xmm8
 0.00 :       1353b7:       movaps %xmm8,0x60(%rdi)
 0.00 :       1353bc:       palignr $0x3,%xmm6,%xmm7
 0.00 :       1353c2:       movaps %xmm7,0x50(%rdi)
```

```
 0.00 :         1353c6:       palignr $0x3,%xmm5,%xmm6
 0.00 :         1353cc:       movaps %xmm6,0x40(%rdi)
 0.00 :         1353d0:       palignr $0x3,%xmm4,%xmm5
 0.00 :         1353d6:       movaps %xmm5,0x30(%rdi)
 0.00 :         1353da:       palignr $0x3,%xmm3,%xmm4
 0.00 :         1353e0:       movaps %xmm4,0x20(%rdi)
 0.00 :         1353e4:       palignr $0x3,%xmm2,%xmm3
 0.00 :         1353ea:       movaps %xmm3,0x10(%rdi)
 0.00 :         1353ee:       palignr $0x3,%xmm1,%xmm2
 0.00 :         1353f4:       movaps %xmm2,(%rdi)
 0.00 :         1353f7:       lea    0x80(%rdi),%rdi
 0.00 :         1353fe:       jae    135370 <__memcpy_ssse3_back+0x520>
 0.00 :         135404:       movdqu %xmm0,(%r8)
 0.00 :         135409:       add    $0x80,%rdx
 0.00 :         135410:       add    %rdx,%rdi
 0.00 :         135413:       add    %rdx,%rsi
 0.00 :         135416:       lea    0x3d013(%rip),%r11        # 172430 <null+0xbe0>
 0.00 :         13541d:       movslq (%r11,%rdx,4),%rdx
 0.00 :         135421:       lea    (%r11,%rdx,1),%rdx
 0.00 :         135425:       jmpq   *%rdx
 0.00 :         135427:       ud2
 0.00 :         135429:       nopl   0x0(%rax)
 0.00 :         135430:       movaps -0x3(%rsi),%xmm1
 0.00 :         135434:       movaps -0x13(%rsi),%xmm2
 0.00 :         135438:       palignr $0x3,%xmm2,%xmm1
 0.00 :         13543e:       movaps %xmm1,-0x10(%rdi)
 0.00 :         135442:       movaps -0x23(%rsi),%xmm3
 0.00 :         135446:       palignr $0x3,%xmm3,%xmm2
 0.00 :         13544c:       movaps %xmm2,-0x20(%rdi)
 0.00 :         135450:       movaps -0x33(%rsi),%xmm4
 0.00 :         135454:       palignr $0x3,%xmm4,%xmm3
 0.00 :         13545a:       movaps %xmm3,-0x30(%rdi)
 0.00 :         13545e:       movaps -0x43(%rsi),%xmm5
 0.00 :         135462:       palignr $0x3,%xmm5,%xmm4
 0.00 :         135468:       movaps %xmm4,-0x40(%rdi)
 0.00 :         13546c:       movaps -0x53(%rsi),%xmm6
 0.00 :         135470:       palignr $0x3,%xmm6,%xmm5
 0.00 :         135476:       movaps %xmm5,-0x50(%rdi)
 0.00 :         13547a:       movaps -0x63(%rsi),%xmm7
 0.00 :         13547e:       palignr $0x3,%xmm7,%xmm6
 0.00 :         135484:       movaps %xmm6,-0x60(%rdi)
 0.00 :         135488:       movaps -0x73(%rsi),%xmm8
 0.00 :         13548d:       palignr $0x3,%xmm8,%xmm7
 0.00 :         135494:       movaps %xmm7,-0x70(%rdi)
 0.00 :         135498:       movaps -0x83(%rsi),%xmm9
 0.00 :         1354a0:       palignr $0x3,%xmm9,%xmm8
 0.00 :         1354a7:       movaps %xmm8,-0x80(%rdi)
 0.00 :         1354ac:       sub    $0x80,%rdx
 0.00 :         1354b3:       lea    -0x80(%rdi),%rdi
 0.00 :         1354b7:       lea    -0x80(%rsi),%rsi
 0.00 :         1354bb:       jae    135430 <__memcpy_ssse3_back+0x5e0>
 0.00 :         1354c1:       movdqu %xmm0,(%r8)
 0.00 :         1354c6:       add    $0x80,%rdx
 0.00 :         1354cd:       sub    %rdx,%rdi
 0.00 :         1354d0:       sub    %rdx,%rsi
 0.00 :         1354d3:       lea    0x3cd16(%rip),%r11        # 1721f0 <null+0x9a0>
 0.00 :         1354da:       movslq (%r11,%rdx,4),%rdx
 0.00 :         1354de:       lea    (%r11,%rdx,1),%rdx
 0.00 :         1354e2:       jmpq   *%rdx
 0.00 :         1354e4:       ud2
 0.00 :         1354e6:       nopw   %cs:0x0(%rax,%rax,1)
 0.00 :         1354f0:       sub    $0x80,%rdx
 0.00 :         1354f7:       movaps -0x4(%rsi),%xmm1
 0.00 :         1354fb:       movaps 0xc(%rsi),%xmm2
```

```
 0.00 :         1354ff:       movaps 0x1c(%rsi),%xmm3
 0.00 :         135503:       movaps 0x2c(%rsi),%xmm4
 0.00 :         135507:       movaps 0x3c(%rsi),%xmm5
 0.00 :         13550b:       movaps 0x4c(%rsi),%xmm6
 0.00 :         13550f:       movaps 0x5c(%rsi),%xmm7
 0.00 :         135513:       movaps 0x6c(%rsi),%xmm8
 0.00 :         135518:       movaps 0x7c(%rsi),%xmm9
 0.00 :         13551d:       lea    0x80(%rsi),%rsi
 0.00 :         135524:       palignr $0x4,%xmm8,%xmm9
 0.00 :         13552b:       movaps %xmm9,0x70(%rdi)
 0.00 :         135530:       palignr $0x4,%xmm7,%xmm8
 1.41 :         135537:       movaps %xmm8,0x60(%rdi)
 0.00 :         13553c:       palignr $0x4,%xmm6,%xmm7
 0.00 :         135542:       movaps %xmm7,0x50(%rdi)
 0.00 :         135546:       palignr $0x4,%xmm5,%xmm6
 0.00 :         13554c:       movaps %xmm6,0x40(%rdi)
 0.00 :         135550:       palignr $0x4,%xmm4,%xmm5
 0.00 :         135556:       movaps %xmm5,0x30(%rdi)
 0.00 :         13555a:       palignr $0x4,%xmm3,%xmm4
 0.00 :         135560:       movaps %xmm4,0x20(%rdi)
 0.00 :         135564:       palignr $0x4,%xmm2,%xmm3
 0.00 :         13556a:       movaps %xmm3,0x10(%rdi)
 1.41 :         13556e:       palignr $0x4,%xmm1,%xmm2
 0.00 :         135574:       movaps %xmm2,(%rdi)
 0.00 :         135577:       lea    0x80(%rdi),%rdi
 0.00 :         13557e:       jae    1354f0 <__memcpy_ssse3_back+0x6a0>
 0.00 :         135584:       movdqu %xmm0,(%r8)
 0.00 :         135589:       add    $0x80,%rdx
 0.00 :         135590:       add    %rdx,%rdi
 0.00 :         135593:       add    %rdx,%rsi
 0.00 :         135596:       lea    0x3ce93(%rip),%r11        # 172430 <null+0xbe0>
 0.00 :         13559d:       movslq (%r11,%rdx,4),%rdx
 0.00 :         1355a1:       lea    (%r11,%rdx,1),%rdx
 0.00 :         1355a5:       jmpq   *%rdx
 0.00 :         1355a7:       ud2
 0.00 :         1355a9:       nopl   0x0(%rax)
 0.00 :         1355b0:       movaps -0x4(%rsi),%xmm1
 0.00 :         1355b4:       movaps -0x14(%rsi),%xmm2
 0.00 :         1355b8:       palignr $0x4,%xmm2,%xmm1
 0.00 :         1355be:       movaps %xmm1,-0x10(%rdi)
 0.00 :         1355c2:       movaps -0x24(%rsi),%xmm3
 0.00 :         1355c6:       palignr $0x4,%xmm3,%xmm2
 0.00 :         1355cc:       movaps %xmm2,-0x20(%rdi)
 0.00 :         1355d0:       movaps -0x34(%rsi),%xmm4
 0.00 :         1355d4:       palignr $0x4,%xmm4,%xmm3
 0.00 :         1355da:       movaps %xmm3,-0x30(%rdi)
 0.00 :         1355de:       movaps -0x44(%rsi),%xmm5
 0.00 :         1355e2:       palignr $0x4,%xmm5,%xmm4
 0.00 :         1355e8:       movaps %xmm4,-0x40(%rdi)
 0.00 :         1355ec:       movaps -0x54(%rsi),%xmm6
 0.00 :         1355f0:       palignr $0x4,%xmm6,%xmm5
 0.00 :         1355f6:       movaps %xmm5,-0x50(%rdi)
 0.00 :         1355fa:       movaps -0x64(%rsi),%xmm7
 0.00 :         1355fe:       palignr $0x4,%xmm7,%xmm6
 0.00 :         135604:       movaps %xmm6,-0x60(%rdi)
 0.00 :         135608:       movaps -0x74(%rsi),%xmm8
 0.00 :         13560d:       palignr $0x4,%xmm8,%xmm7
 0.00 :         135614:       movaps %xmm7,-0x70(%rdi)
 0.00 :         135618:       movaps -0x84(%rsi),%xmm9
 0.00 :         135620:       palignr $0x4,%xmm9,%xmm8
 0.00 :         135627:       movaps %xmm8,-0x80(%rdi)
 0.00 :         13562c:       sub    $0x80,%rdx
 0.00 :         135633:       lea    -0x80(%rdi),%rdi
 0.00 :         135637:       lea    -0x80(%rsi),%rsi
```

```
  0.00 :         13563b:       jae    1355b0 <__memcpy_ssse3_back+0x760>
  0.00 :         135641:       movdqu %xmm0,(%r8)
  0.00 :         135646:       add    $0x80,%rdx
  0.00 :         13564d:       sub    %rdx,%rdi
  0.00 :         135650:       sub    %rdx,%rsi
  0.00 :         135653:       lea    0x3cb96(%rip),%r11        # 1721f0 <null+0x9a0>
  0.00 :         13565a:       movslq (%r11,%rdx,4),%rdx
  0.00 :         13565e:       lea    (%r11,%rdx,1),%rdx
  0.00 :         135662:       jmpq   *%rdx
  0.00 :         135664:       ud2
  0.00 :         135666:       nopw   %cs:0x0(%rax,%rax,1)
  0.00 :         135670:       sub    $0x80,%rdx
  0.00 :         135677:       movaps -0x5(%rsi),%xmm1
  0.00 :         13567b:       movaps 0xb(%rsi),%xmm2
  0.00 :         13567f:       movaps 0x1b(%rsi),%xmm3
  0.00 :         135683:       movaps 0x2b(%rsi),%xmm4
  0.00 :         135687:       movaps 0x3b(%rsi),%xmm5
  0.00 :         13568b:       movaps 0x4b(%rsi),%xmm6
  0.00 :         13568f:       movaps 0x5b(%rsi),%xmm7
  0.00 :         135693:       movaps 0x6b(%rsi),%xmm8
  0.00 :         135698:       movaps 0x7b(%rsi),%xmm9
  0.00 :         13569d:       lea    0x80(%rsi),%rsi
  0.00 :         1356a4:       palignr $0x5,%xmm8,%xmm9
  0.00 :         1356ab:       movaps %xmm9,0x70(%rdi)
  0.00 :         1356b0:       palignr $0x5,%xmm7,%xmm8
  0.00 :         1356b7:       movaps %xmm8,0x60(%rdi)
  0.00 :         1356bc:       palignr $0x5,%xmm6,%xmm7
  0.00 :         1356c2:       movaps %xmm7,0x50(%rdi)
  0.00 :         1356c6:       palignr $0x5,%xmm5,%xmm6
  0.00 :         1356cc:       movaps %xmm6,0x40(%rdi)
  0.00 :         1356d0:       palignr $0x5,%xmm4,%xmm5
  0.00 :         1356d6:       movaps %xmm5,0x30(%rdi)
  0.00 :         1356da:       palignr $0x5,%xmm3,%xmm4
  0.00 :         1356e0:       movaps %xmm4,0x20(%rdi)
  0.00 :         1356e4:       palignr $0x5,%xmm2,%xmm3
  0.00 :         1356ea:       movaps %xmm3,0x10(%rdi)
  0.00 :         1356ee:       palignr $0x5,%xmm1,%xmm2
  0.00 :         1356f4:       movaps %xmm2,(%rdi)
  0.00 :         1356f7:       lea    0x80(%rdi),%rdi
  0.00 :         1356fe:       jae    135670 <__memcpy_ssse3_back+0x820>
  0.00 :         135704:       movdqu %xmm0,(%r8)
  0.00 :         135709:       add    $0x80,%rdx
  0.00 :         135710:       add    %rdx,%rdi
  0.00 :         135713:       add    %rdx,%rsi
  0.00 :         135716:       lea    0x3cd13(%rip),%r11        # 172430 <null+0xbe0>
  0.00 :         13571d:       movslq (%r11,%rdx,4),%rdx
  0.00 :         135721:       lea    (%r11,%rdx,1),%rdx
  0.00 :         135725:       jmpq   *%rdx
  0.00 :         135727:       ud2
  0.00 :         135729:       nopl   0x0(%rax)
  0.00 :         135730:       movaps -0x5(%rsi),%xmm1
  0.00 :         135734:       movaps -0x15(%rsi),%xmm2
  0.00 :         135738:       palignr $0x5,%xmm2,%xmm1
  0.00 :         13573e:       movaps %xmm1,-0x10(%rdi)
  0.00 :         135742:       movaps -0x25(%rsi),%xmm3
  0.00 :         135746:       palignr $0x5,%xmm3,%xmm2
  0.00 :         13574c:       movaps %xmm2,-0x20(%rdi)
  0.00 :         135750:       movaps -0x35(%rsi),%xmm4
  0.00 :         135754:       palignr $0x5,%xmm4,%xmm3
  0.00 :         13575a:       movaps %xmm3,-0x30(%rdi)
  0.00 :         13575e:       movaps -0x45(%rsi),%xmm5
  0.00 :         135762:       palignr $0x5,%xmm5,%xmm4
  0.00 :         135768:       movaps %xmm4,-0x40(%rdi)
  0.00 :         13576c:       movaps -0x55(%rsi),%xmm6
```

```
 0.00 :         135770:       palignr $0x5,%xmm6,%xmm5
 0.00 :         135776:       movaps %xmm5,-0x50(%rdi)
 0.00 :         13577a:       movaps -0x65(%rsi),%xmm7
 0.00 :         13577e:       palignr $0x5,%xmm7,%xmm6
 0.00 :         135784:       movaps %xmm6,-0x60(%rdi)
 0.00 :         135788:       movaps -0x75(%rsi),%xmm8
 0.00 :         13578d:       palignr $0x5,%xmm8,%xmm7
 0.00 :         135794:       movaps %xmm7,-0x70(%rdi)
 0.00 :         135798:       movaps -0x85(%rsi),%xmm9
 0.00 :         1357a0:       palignr $0x5,%xmm9,%xmm8
 0.00 :         1357a7:       movaps %xmm8,-0x80(%rdi)
 0.00 :         1357ac:       sub    $0x80,%rdx
 0.00 :         1357b3:       lea    -0x80(%rdi),%rdi
 0.00 :         1357b7:       lea    -0x80(%rsi),%rsi
 0.00 :         1357bb:       jae    135730 <__memcpy_ssse3_back+0x8e0>
 0.00 :         1357c1:       movdqu %xmm0,(%r8)
 0.00 :         1357c6:       add    $0x80,%rdx
 0.00 :         1357cd:       sub    %rdx,%rdi
 0.00 :         1357d0:       sub    %rdx,%rsi
 0.00 :         1357d3:       lea    0x3ca16(%rip),%r11        # 1721f0 <null+0x9a0>
 0.00 :         1357da:       movslq (%r11,%rdx,4),%rdx
 0.00 :         1357de:       lea    (%r11,%rdx,1),%rdx
 0.00 :         1357e2:       jmpq   *%rdx
 0.00 :         1357e4:       ud2
 0.00 :         1357e6:       nopw   %cs:0x0(%rax,%rax,1)
 0.00 :         1357f0:       sub    $0x80,%rdx
 0.00 :         1357f7:       movaps -0x6(%rsi),%xmm1
 0.00 :         1357fb:       movaps 0xa(%rsi),%xmm2
 0.00 :         1357ff:       movaps 0x1a(%rsi),%xmm3
 0.00 :         135803:       movaps 0x2a(%rsi),%xmm4
 0.00 :         135807:       movaps 0x3a(%rsi),%xmm5
 1.41 :         13580b:       movaps 0x4a(%rsi),%xmm6
 0.00 :         13580f:       movaps 0x5a(%rsi),%xmm7
 0.00 :         135813:       movaps 0x6a(%rsi),%xmm8
 0.00 :         135818:       movaps 0x7a(%rsi),%xmm9
 1.41 :         13581d:       lea    0x80(%rsi),%rsi
 0.00 :         135824:       palignr $0x6,%xmm8,%xmm9
 0.00 :         13582b:       movaps %xmm9,0x70(%rdi)
 0.00 :         135830:       palignr $0x6,%xmm7,%xmm8
 0.00 :         135837:       movaps %xmm8,0x60(%rdi)
 1.41 :         13583c:       palignr $0x6,%xmm6,%xmm7
 0.00 :         135842:       movaps %xmm7,0x50(%rdi)
 0.00 :         135846:       palignr $0x6,%xmm5,%xmm6
 0.00 :         13584c:       movaps %xmm6,0x40(%rdi)
 0.00 :         135850:       palignr $0x6,%xmm4,%xmm5
 0.00 :         135856:       movaps %xmm5,0x30(%rdi)
 0.00 :         13585a:       palignr $0x6,%xmm3,%xmm4
 0.00 :         135860:       movaps %xmm4,0x20(%rdi)
 0.00 :         135864:       palignr $0x6,%xmm2,%xmm3
 0.00 :         13586a:       movaps %xmm3,0x10(%rdi)
 0.00 :         13586e:       palignr $0x6,%xmm1,%xmm2
 0.00 :         135874:       movaps %xmm2,(%rdi)
 0.00 :         135877:       lea    0x80(%rdi),%rdi
 0.00 :         13587e:       jae    1357f0 <__memcpy_ssse3_back+0x9a0>
 0.00 :         135884:       movdqu %xmm0,(%r8)
 0.00 :         135889:       add    $0x80,%rdx
 0.00 :         135890:       add    %rdx,%rdi
 0.00 :         135893:       add    %rdx,%rsi
 0.00 :         135896:       lea    0x3cb93(%rip),%r11        # 172430 <null+0xbe0>
 0.00 :         13589d:       movslq (%r11,%rdx,4),%rdx
 0.00 :         1358a1:       lea    (%r11,%rdx,1),%rdx
 0.00 :         1358a5:       jmpq   *%rdx
 0.00 :         1358a7:       ud2
 0.00 :         1358a9:       nopl   0x0(%rax)
```

```
 0.00 :         1358b0:       movaps -0x6(%rsi),%xmm1
 0.00 :         1358b4:       movaps -0x16(%rsi),%xmm2
 0.00 :         1358b8:       palignr $0x6,%xmm2,%xmm1
 0.00 :         1358be:       movaps %xmm1,-0x10(%rdi)
 0.00 :         1358c2:       movaps -0x26(%rsi),%xmm3
 0.00 :         1358c6:       palignr $0x6,%xmm3,%xmm2
 0.00 :         1358cc:       movaps %xmm2,-0x20(%rdi)
 0.00 :         1358d0:       movaps -0x36(%rsi),%xmm4
 0.00 :         1358d4:       palignr $0x6,%xmm4,%xmm3
 0.00 :         1358da:       movaps %xmm3,-0x30(%rdi)
 0.00 :         1358de:       movaps -0x46(%rsi),%xmm5
 0.00 :         1358e2:       palignr $0x6,%xmm5,%xmm4
 0.00 :         1358e8:       movaps %xmm4,-0x40(%rdi)
 0.00 :         1358ec:       movaps -0x56(%rsi),%xmm6
 0.00 :         1358f0:       palignr $0x6,%xmm6,%xmm5
 0.00 :         1358f6:       movaps %xmm5,-0x50(%rdi)
 0.00 :         1358fa:       movaps -0x66(%rsi),%xmm7
 0.00 :         1358fe:       palignr $0x6,%xmm7,%xmm6
 0.00 :         135904:       movaps %xmm6,-0x60(%rdi)
 0.00 :         135908:       movaps -0x76(%rsi),%xmm8
 0.00 :         13590d:       palignr $0x6,%xmm8,%xmm7
 0.00 :         135914:       movaps %xmm7,-0x70(%rdi)
 0.00 :         135918:       movaps -0x86(%rsi),%xmm9
 0.00 :         135920:       palignr $0x6,%xmm9,%xmm8
 0.00 :         135927:       movaps %xmm8,-0x80(%rdi)
 0.00 :         13592c:       sub    $0x80,%rdx
 0.00 :         135933:       lea    -0x80(%rdi),%rdi
 0.00 :         135937:       lea    -0x80(%rsi),%rsi
 0.00 :         13593b:       jae    1358b0 <__memcpy_ssse3_back+0xa60>
 0.00 :         135941:       movdqu %xmm0,(%r8)
 0.00 :         135946:       add    $0x80,%rdx
 0.00 :         13594d:       sub    %rdx,%rdi
 0.00 :         135950:       sub    %rdx,%rsi
 0.00 :         135953:       lea    0x3c896(%rip),%r11        # 1721f0 <null+0x9a0>
 0.00 :         13595a:       movslq (%r11,%rdx,4),%rdx
 0.00 :         13595e:       lea    (%r11,%rdx,1),%rdx
 0.00 :         135962:       jmpq   *%rdx
 0.00 :         135964:       ud2
 0.00 :         135966:       nopw   %cs:0x0(%rax,%rax,1)
 0.00 :         135970:       sub    $0x80,%rdx
 0.00 :         135977:       movaps -0x7(%rsi),%xmm1
 0.00 :         13597b:       movaps 0x9(%rsi),%xmm2
 0.00 :         13597f:       movaps 0x19(%rsi),%xmm3
 0.00 :         135983:       movaps 0x29(%rsi),%xmm4
 0.00 :         135987:       movaps 0x39(%rsi),%xmm5
 0.00 :         13598b:       movaps 0x49(%rsi),%xmm6
 0.00 :         13598f:       movaps 0x59(%rsi),%xmm7
 1.41 :         135993:       movaps 0x69(%rsi),%xmm8
 0.00 :         135998:       movaps 0x79(%rsi),%xmm9
 0.00 :         13599d:       lea    0x80(%rsi),%rsi
 0.00 :         1359a4:       palignr $0x7,%xmm8,%xmm9
 1.41 :         1359ab:       movaps %xmm9,0x70(%rdi)
 0.00 :         1359b0:       palignr $0x7,%xmm7,%xmm8
 0.00 :         1359b7:       movaps %xmm8,0x60(%rdi)
 0.00 :         1359bc:       palignr $0x7,%xmm6,%xmm7
 0.00 :         1359c2:       movaps %xmm7,0x50(%rdi)
 0.00 :         1359c6:       palignr $0x7,%xmm5,%xmm6
 0.00 :         1359cc:       movaps %xmm6,0x40(%rdi)
 0.00 :         1359d0:       palignr $0x7,%xmm4,%xmm5
 0.00 :         1359d6:       movaps %xmm5,0x30(%rdi)
 0.00 :         1359da:       palignr $0x7,%xmm3,%xmm4
 0.00 :         1359e0:       movaps %xmm4,0x20(%rdi)
 0.00 :         1359e4:       palignr $0x7,%xmm2,%xmm3
 0.00 :         1359ea:       movaps %xmm3,0x10(%rdi)
```

```
 0.00 :        1359ee:       palignr $0x7,%xmm1,%xmm2
 0.00 :        1359f4:       movaps %xmm2,(%rdi)
 1.41 :        1359f7:       lea    0x80(%rdi),%rdi
 0.00 :        1359fe:       jae    135970 <__memcpy_ssse3_back+0xb20>
 0.00 :        135a04:       movdqu %xmm0,(%r8)
 0.00 :        135a09:       add    $0x80,%rdx
 0.00 :        135a10:       add    %rdx,%rdi
 0.00 :        135a13:       add    %rdx,%rsi
 0.00 :        135a16:       lea    0x3ca13(%rip),%r11        # 172430 <null+0xbe0>
 0.00 :        135a1d:       movslq (%r11,%rdx,4),%rdx
 0.00 :        135a21:       lea    (%r11,%rdx,1),%rdx
 0.00 :        135a25:       jmpq   *%rdx
 0.00 :        135a27:       ud2
 0.00 :        135a29:       nopl   0x0(%rax)
 0.00 :        135a30:       movaps -0x7(%rsi),%xmm1
 0.00 :        135a34:       movaps -0x17(%rsi),%xmm2
 0.00 :        135a38:       palignr $0x7,%xmm2,%xmm1
 0.00 :        135a3e:       movaps %xmm1,-0x10(%rdi)
 0.00 :        135a42:       movaps -0x27(%rsi),%xmm3
 0.00 :        135a46:       palignr $0x7,%xmm3,%xmm2
 0.00 :        135a4c:       movaps %xmm2,-0x20(%rdi)
 0.00 :        135a50:       movaps -0x37(%rsi),%xmm4
 0.00 :        135a54:       palignr $0x7,%xmm4,%xmm3
 0.00 :        135a5a:       movaps %xmm3,-0x30(%rdi)
 0.00 :        135a5e:       movaps -0x47(%rsi),%xmm5
 0.00 :        135a62:       palignr $0x7,%xmm5,%xmm4
 0.00 :        135a68:       movaps %xmm4,-0x40(%rdi)
 0.00 :        135a6c:       movaps -0x57(%rsi),%xmm6
 0.00 :        135a70:       palignr $0x7,%xmm6,%xmm5
 0.00 :        135a76:       movaps %xmm5,-0x50(%rdi)
 0.00 :        135a7a:       movaps -0x67(%rsi),%xmm7
 0.00 :        135a7e:       palignr $0x7,%xmm7,%xmm6
 0.00 :        135a84:       movaps %xmm6,-0x60(%rdi)
 0.00 :        135a88:       movaps -0x77(%rsi),%xmm8
 0.00 :        135a8d:       palignr $0x7,%xmm8,%xmm7
 0.00 :        135a94:       movaps %xmm7,-0x70(%rdi)
 0.00 :        135a98:       movaps -0x87(%rsi),%xmm9
 0.00 :        135aa0:       palignr $0x7,%xmm9,%xmm8
 0.00 :        135aa7:       movaps %xmm8,-0x80(%rdi)
 0.00 :        135aac:       sub    $0x80,%rdx
 0.00 :        135ab3:       lea    -0x80(%rdi),%rdi
 0.00 :        135ab7:       lea    -0x80(%rsi),%rsi
 0.00 :        135abb:       jae    135a30 <__memcpy_ssse3_back+0xbe0>
 0.00 :        135ac1:       movdqu %xmm0,(%r8)
 0.00 :        135ac6:       add    $0x80,%rdx
 0.00 :        135acd:       sub    %rdx,%rdi
 0.00 :        135ad0:       sub    %rdx,%rsi
 0.00 :        135ad3:       lea    0x3c716(%rip),%r11        # 1721f0 <null+0x9a0>
 0.00 :        135ada:       movslq (%r11,%rdx,4),%rdx
 0.00 :        135ade:       lea    (%r11,%rdx,1),%rdx
 0.00 :        135ae2:       jmpq   *%rdx
 0.00 :        135ae4:       ud2
 0.00 :        135ae6:       nopw   %cs:0x0(%rax,%rax,1)
 0.00 :        135af0:       sub    $0x80,%rdx
 0.00 :        135af7:       movaps -0x8(%rsi),%xmm1
 0.00 :        135afb:       movaps 0x8(%rsi),%xmm2
 0.00 :        135aff:       movaps 0x18(%rsi),%xmm3
 0.00 :        135b03:       movaps 0x28(%rsi),%xmm4
 0.00 :        135b07:       movaps 0x38(%rsi),%xmm5
 0.00 :        135b0b:       movaps 0x48(%rsi),%xmm6
 0.00 :        135b0f:       movaps 0x58(%rsi),%xmm7
 0.00 :        135b13:       movaps 0x68(%rsi),%xmm8
 2.82 :        135b18:       movaps 0x78(%rsi),%xmm9
 0.00 :        135b1d:       lea    0x80(%rsi),%rsi
```

```
 1.41 :         135b24:       palignr $0x8,%xmm8,%xmm9
 0.00 :         135b2b:       movaps %xmm9,0x70(%rdi)
 0.00 :         135b30:       palignr $0x8,%xmm7,%xmm8
 0.00 :         135b37:       movaps %xmm8,0x60(%rdi)
 0.00 :         135b3c:       palignr $0x8,%xmm6,%xmm7
 0.00 :         135b42:       movaps %xmm7,0x50(%rdi)
 1.41 :         135b46:       palignr $0x8,%xmm5,%xmm6
 0.00 :         135b4c:       movaps %xmm6,0x40(%rdi)
 0.00 :         135b50:       palignr $0x8,%xmm4,%xmm5
 1.41 :         135b56:       movaps %xmm5,0x30(%rdi)
 0.00 :         135b5a:       palignr $0x8,%xmm3,%xmm4
 0.00 :         135b60:       movaps %xmm4,0x20(%rdi)
 0.00 :         135b64:       palignr $0x8,%xmm2,%xmm3
 0.00 :         135b6a:       movaps %xmm3,0x10(%rdi)
 0.00 :         135b6e:       palignr $0x8,%xmm1,%xmm2
 0.00 :         135b74:       movaps %xmm2,(%rdi)
 0.00 :         135b77:       lea    0x80(%rdi),%rdi
 0.00 :         135b7e:       jae    135af0 <__memcpy_ssse3_back+0xca0>
 0.00 :         135b84:       movdqu %xmm0,(%r8)
 0.00 :         135b89:       add    $0x80,%rdx
 0.00 :         135b90:       add    %rdx,%rdi
 0.00 :         135b93:       add    %rdx,%rsi
 0.00 :         135b96:       lea    0x3c893(%rip),%r11        # 172430 <null+0xbe0>
 0.00 :         135b9d:       movslq (%r11,%rdx,4),%rdx
 0.00 :         135ba1:       lea    (%r11,%rdx,1),%rdx
 0.00 :         135ba5:       jmpq   *%rdx
 0.00 :         135ba7:       ud2
 0.00 :         135ba9:       nopl   0x0(%rax)
 0.00 :         135bb0:       movaps -0x8(%rsi),%xmm1
 0.00 :         135bb4:       movaps -0x18(%rsi),%xmm2
 0.00 :         135bb8:       palignr $0x8,%xmm2,%xmm1
 0.00 :         135bbe:       movaps %xmm1,-0x10(%rdi)
 0.00 :         135bc2:       movaps -0x28(%rsi),%xmm3
 0.00 :         135bc6:       palignr $0x8,%xmm3,%xmm2
 0.00 :         135bcc:       movaps %xmm2,-0x20(%rdi)
 0.00 :         135bd0:       movaps -0x38(%rsi),%xmm4
 0.00 :         135bd4:       palignr $0x8,%xmm4,%xmm3
 0.00 :         135bda:       movaps %xmm3,-0x30(%rdi)
 0.00 :         135bde:       movaps -0x48(%rsi),%xmm5
 0.00 :         135be2:       palignr $0x8,%xmm5,%xmm4
 0.00 :         135be8:       movaps %xmm4,-0x40(%rdi)
 0.00 :         135bec:       movaps -0x58(%rsi),%xmm6
 0.00 :         135bf0:       palignr $0x8,%xmm6,%xmm5
 0.00 :         135bf6:       movaps %xmm5,-0x50(%rdi)
 0.00 :         135bfa:       movaps -0x68(%rsi),%xmm7
 0.00 :         135bfe:       palignr $0x8,%xmm7,%xmm6
 0.00 :         135c04:       movaps %xmm6,-0x60(%rdi)
 0.00 :         135c08:       movaps -0x78(%rsi),%xmm8
 0.00 :         135c0d:       palignr $0x8,%xmm8,%xmm7
 0.00 :         135c14:       movaps %xmm7,-0x70(%rdi)
 0.00 :         135c18:       movaps -0x88(%rsi),%xmm9
 0.00 :         135c20:       palignr $0x8,%xmm9,%xmm8
 0.00 :         135c27:       movaps %xmm8,-0x80(%rdi)
 0.00 :         135c2c:       sub    $0x80,%rdx
 0.00 :         135c33:       lea    -0x80(%rdi),%rdi
 0.00 :         135c37:       lea    -0x80(%rsi),%rsi
 0.00 :         135c3b:       jae    135bb0 <__memcpy_ssse3_back+0xd60>
 0.00 :         135c41:       movdqu %xmm0,(%r8)
 0.00 :         135c46:       add    $0x80,%rdx
 0.00 :         135c4d:       sub    %rdx,%rdi
 0.00 :         135c50:       sub    %rdx,%rsi
 0.00 :         135c53:       lea    0x3c596(%rip),%r11        # 1721f0 <null+0x9a0>
 0.00 :         135c5a:       movslq (%r11,%rdx,4),%rdx
 0.00 :         135c5e:       lea    (%r11,%rdx,1),%rdx
```

```
 0.00 :         135c62:       jmpq   *%rdx
 0.00 :         135c64:       ud2
 0.00 :         135c66:       nopw   %cs:0x0(%rax,%rax,1)
 0.00 :         135c70:       sub    $0x80,%rdx
 0.00 :         135c77:       movaps -0x9(%rsi),%xmm1
 0.00 :         135c7b:       movaps 0x7(%rsi),%xmm2
 0.00 :         135c7f:       movaps 0x17(%rsi),%xmm3
 0.00 :         135c83:       movaps 0x27(%rsi),%xmm4
 0.00 :         135c87:       movaps 0x37(%rsi),%xmm5
 0.00 :         135c8b:       movaps 0x47(%rsi),%xmm6
 0.00 :         135c8f:       movaps 0x57(%rsi),%xmm7
 0.00 :         135c93:       movaps 0x67(%rsi),%xmm8
 0.00 :         135c98:       movaps 0x77(%rsi),%xmm9
 0.00 :         135c9d:       lea    0x80(%rsi),%rsi
 0.00 :         135ca4:       palignr $0x9,%xmm8,%xmm9
 0.00 :         135cab:       movaps %xmm9,0x70(%rdi)
 0.00 :         135cb0:       palignr $0x9,%xmm7,%xmm8
 0.00 :         135cb7:       movaps %xmm8,0x60(%rdi)
 0.00 :         135cbc:       palignr $0x9,%xmm6,%xmm7
 0.00 :         135cc2:       movaps %xmm7,0x50(%rdi)
 0.00 :         135cc6:       palignr $0x9,%xmm5,%xmm6
 0.00 :         135ccc:       movaps %xmm6,0x40(%rdi)
 0.00 :         135cd0:       palignr $0x9,%xmm4,%xmm5
 0.00 :         135cd6:       movaps %xmm5,0x30(%rdi)
 0.00 :         135cda:       palignr $0x9,%xmm3,%xmm4
 0.00 :         135ce0:       movaps %xmm4,0x20(%rdi)
 0.00 :         135ce4:       palignr $0x9,%xmm2,%xmm3
 0.00 :         135cea:       movaps %xmm3,0x10(%rdi)
 0.00 :         135cee:       palignr $0x9,%xmm1,%xmm2
 0.00 :         135cf4:       movaps %xmm2,(%rdi)
 0.00 :         135cf7:       lea    0x80(%rdi),%rdi
 0.00 :         135cfe:       jae    135c70 <__memcpy_ssse3_back+0xe20>
 0.00 :         135d04:       movdqu %xmm0,(%r8)
 0.00 :         135d09:       add    $0x80,%rdx
 0.00 :         135d10:       add    %rdx,%rdi
 0.00 :         135d13:       add    %rdx,%rsi
 0.00 :         135d16:       lea    0x3c713(%rip),%r11        # 172430 <null+0xbe0>
 0.00 :         135d1d:       movslq (%r11,%rdx,4),%rdx
 0.00 :         135d21:       lea    (%r11,%rdx,1),%rdx
 0.00 :         135d25:       jmpq   *%rdx
 0.00 :         135d27:       ud2
 0.00 :         135d29:       nopl   0x0(%rax)
 0.00 :         135d30:       movaps -0x9(%rsi),%xmm1
 0.00 :         135d34:       movaps -0x19(%rsi),%xmm2
 0.00 :         135d38:       palignr $0x9,%xmm2,%xmm1
 0.00 :         135d3e:       movaps %xmm1,-0x10(%rdi)
 0.00 :         135d42:       movaps -0x29(%rsi),%xmm3
 0.00 :         135d46:       palignr $0x9,%xmm3,%xmm2
 0.00 :         135d4c:       movaps %xmm2,-0x20(%rdi)
 0.00 :         135d50:       movaps -0x39(%rsi),%xmm4
 0.00 :         135d54:       palignr $0x9,%xmm4,%xmm3
 0.00 :         135d5a:       movaps %xmm3,-0x30(%rdi)
 0.00 :         135d5e:       movaps -0x49(%rsi),%xmm5
 0.00 :         135d62:       palignr $0x9,%xmm5,%xmm4
 1.41 :         135d68:       movaps %xmm4,-0x40(%rdi)
 0.00 :         135d6c:       movaps -0x59(%rsi),%xmm6
 0.00 :         135d70:       palignr $0x9,%xmm6,%xmm5
 0.00 :         135d76:       movaps %xmm5,-0x50(%rdi)
 0.00 :         135d7a:       movaps -0x69(%rsi),%xmm7
 0.00 :         135d7e:       palignr $0x9,%xmm7,%xmm6
 0.00 :         135d84:       movaps %xmm6,-0x60(%rdi)
 0.00 :         135d88:       movaps -0x79(%rsi),%xmm8
 0.00 :         135d8d:       palignr $0x9,%xmm8,%xmm7
 0.00 :         135d94:       movaps %xmm7,-0x70(%rdi)
```

```
 0.00 :         135d98:       movaps -0x89(%rsi),%xmm9
 0.00 :         135da0:       palignr $0x9,%xmm9,%xmm8
 0.00 :         135da7:       movaps %xmm8,-0x80(%rdi)
 0.00 :         135dac:       sub    $0x80,%rdx
 0.00 :         135db3:       lea    -0x80(%rdi),%rdi
 0.00 :         135db7:       lea    -0x80(%rsi),%rsi
 0.00 :         135dbb:       jae    135d30 <__memcpy_ssse3_back+0xee0>
 0.00 :         135dc1:       movdqu %xmm0,(%r8)
 0.00 :         135dc6:       add    $0x80,%rdx
 0.00 :         135dcd:       sub    %rdx,%rdi
 0.00 :         135dd0:       sub    %rdx,%rsi
 0.00 :         135dd3:       lea    0x3c416(%rip),%r11        # 1721f0 <null+0x9a0>
 0.00 :         135dda:       movslq (%r11,%rdx,4),%rdx
 0.00 :         135dde:       lea    (%r11,%rdx,1),%rdx
 0.00 :         135de2:       jmpq   *%rdx
 0.00 :         135de4:       ud2
 0.00 :         135de6:       nopw   %cs:0x0(%rax,%rax,1)
 0.00 :         135df0:       sub    $0x80,%rdx
 0.00 :         135df7:       movaps -0xa(%rsi),%xmm1
 0.00 :         135dfb:       movaps 0x6(%rsi),%xmm2
 0.00 :         135dff:       movaps 0x16(%rsi),%xmm3
 0.00 :         135e03:       movaps 0x26(%rsi),%xmm4
 0.00 :         135e07:       movaps 0x36(%rsi),%xmm5
 0.00 :         135e0b:       movaps 0x46(%rsi),%xmm6
 0.00 :         135e0f:       movaps 0x56(%rsi),%xmm7
 0.00 :         135e13:       movaps 0x66(%rsi),%xmm8
 0.00 :         135e18:       movaps 0x76(%rsi),%xmm9
 0.00 :         135e1d:       lea    0x80(%rsi),%rsi
 0.00 :         135e24:       palignr $0xa,%xmm8,%xmm9
 0.00 :         135e2b:       movaps %xmm9,0x70(%rdi)
 0.00 :         135e30:       palignr $0xa,%xmm7,%xmm8
 0.00 :         135e37:       movaps %xmm8,0x60(%rdi)
 0.00 :         135e3c:       palignr $0xa,%xmm6,%xmm7
 0.00 :         135e42:       movaps %xmm7,0x50(%rdi)
 0.00 :         135e46:       palignr $0xa,%xmm5,%xmm6
 0.00 :         135e4c:       movaps %xmm6,0x40(%rdi)
 0.00 :         135e50:       palignr $0xa,%xmm4,%xmm5
 0.00 :         135e56:       movaps %xmm5,0x30(%rdi)
 0.00 :         135e5a:       palignr $0xa,%xmm3,%xmm4
 0.00 :         135e60:       movaps %xmm4,0x20(%rdi)
 0.00 :         135e64:       palignr $0xa,%xmm2,%xmm3
 0.00 :         135e6a:       movaps %xmm3,0x10(%rdi)
 0.00 :         135e6e:       palignr $0xa,%xmm1,%xmm2
 0.00 :         135e74:       movaps %xmm2,(%rdi)
 0.00 :         135e77:       lea    0x80(%rdi),%rdi
 0.00 :         135e7e:       jae    135df0 <__memcpy_ssse3_back+0xfa0>
 0.00 :         135e84:       movdqu %xmm0,(%r8)
 0.00 :         135e89:       add    $0x80,%rdx
 0.00 :         135e90:       add    %rdx,%rdi
 0.00 :         135e93:       add    %rdx,%rsi
 0.00 :         135e96:       lea    0x3c593(%rip),%r11        # 172430 <null+0xbe0>
 0.00 :         135e9d:       movslq (%r11,%rdx,4),%rdx
 0.00 :         135ea1:       lea    (%r11,%rdx,1),%rdx
 0.00 :         135ea5:       jmpq   *%rdx
 0.00 :         135ea7:       ud2
 0.00 :         135ea9:       nopl   0x0(%rax)
 0.00 :         135eb0:       movaps -0xa(%rsi),%xmm1
 0.00 :         135eb4:       movaps -0x1a(%rsi),%xmm2
 0.00 :         135eb8:       palignr $0xa,%xmm2,%xmm1
 0.00 :         135ebe:       movaps %xmm1,-0x10(%rdi)
 0.00 :         135ec2:       movaps -0x2a(%rsi),%xmm3
 0.00 :         135ec6:       palignr $0xa,%xmm3,%xmm2
 0.00 :         135ecc:       movaps %xmm2,-0x20(%rdi)
 0.00 :         135ed0:       movaps -0x3a(%rsi),%xmm4
```

```
 0.00 :        135ed4:       palignr $0xa,%xmm4,%xmm3
 0.00 :        135eda:       movaps %xmm3,-0x30(%rdi)
 0.00 :        135ede:       movaps -0x4a(%rsi),%xmm5
 0.00 :        135ee2:       palignr $0xa,%xmm5,%xmm4
 0.00 :        135ee8:       movaps %xmm4,-0x40(%rdi)
 0.00 :        135eec:       movaps -0x5a(%rsi),%xmm6
 0.00 :        135ef0:       palignr $0xa,%xmm6,%xmm5
 0.00 :        135ef6:       movaps %xmm5,-0x50(%rdi)
 0.00 :        135efa:       movaps -0x6a(%rsi),%xmm7
 0.00 :        135efe:       palignr $0xa,%xmm7,%xmm6
 0.00 :        135f04:       movaps %xmm6,-0x60(%rdi)
 0.00 :        135f08:       movaps -0x7a(%rsi),%xmm8
 0.00 :        135f0d:       palignr $0xa,%xmm8,%xmm7
 0.00 :        135f14:       movaps %xmm7,-0x70(%rdi)
 0.00 :        135f18:       movaps -0x8a(%rsi),%xmm9
 0.00 :        135f20:       palignr $0xa,%xmm9,%xmm8
 0.00 :        135f27:       movaps %xmm8,-0x80(%rdi)
 0.00 :        135f2c:       sub    $0x80,%rdx
 0.00 :        135f33:       lea    -0x80(%rdi),%rdi
 0.00 :        135f37:       lea    -0x80(%rsi),%rsi
 0.00 :        135f3b:       jae    135eb0 <__memcpy_ssse3_back+0x1060>
 0.00 :        135f41:       movdqu %xmm0,(%r8)
 0.00 :        135f46:       add    $0x80,%rdx
 0.00 :        135f4d:       sub    %rdx,%rdi
 0.00 :        135f50:       sub    %rdx,%rsi
 0.00 :        135f53:       lea    0x3c296(%rip),%r11        # 1721f0 <null+0x9a0>
 0.00 :        135f5a:       movslq (%r11,%rdx,4),%rdx
 0.00 :        135f5e:       lea    (%r11,%rdx,1),%rdx
 0.00 :        135f62:       jmpq   *%rdx
 0.00 :        135f64:       ud2
 0.00 :        135f66:       nopw   %cs:0x0(%rax,%rax,1)
 0.00 :        135f70:       sub    $0x80,%rdx
 0.00 :        135f77:       movaps -0xb(%rsi),%xmm1
 0.00 :        135f7b:       movaps 0x5(%rsi),%xmm2
 0.00 :        135f7f:       movaps 0x15(%rsi),%xmm3
 0.00 :        135f83:       movaps 0x25(%rsi),%xmm4
 0.00 :        135f87:       movaps 0x35(%rsi),%xmm5
 0.00 :        135f8b:       movaps 0x45(%rsi),%xmm6
 0.00 :        135f8f:       movaps 0x55(%rsi),%xmm7
 0.00 :        135f93:       movaps 0x65(%rsi),%xmm8
 0.00 :        135f98:       movaps 0x75(%rsi),%xmm9
 0.00 :        135f9d:       lea    0x80(%rsi),%rsi
 0.00 :        135fa4:       palignr $0xb,%xmm8,%xmm9
 0.00 :        135fab:       movaps %xmm9,0x70(%rdi)
 0.00 :        135fb0:       palignr $0xb,%xmm7,%xmm8
 0.00 :        135fb7:       movaps %xmm8,0x60(%rdi)
 0.00 :        135fbc:       palignr $0xb,%xmm6,%xmm7
 0.00 :        135fc2:       movaps %xmm7,0x50(%rdi)
 0.00 :        135fc6:       palignr $0xb,%xmm5,%xmm6
 0.00 :        135fcc:       movaps %xmm6,0x40(%rdi)
 1.41 :        135fd0:       palignr $0xb,%xmm4,%xmm5
 0.00 :        135fd6:       movaps %xmm5,0x30(%rdi)
 0.00 :        135fda:       palignr $0xb,%xmm3,%xmm4
 0.00 :        135fe0:       movaps %xmm4,0x20(%rdi)
 0.00 :        135fe4:       palignr $0xb,%xmm2,%xmm3
 0.00 :        135fea:       movaps %xmm3,0x10(%rdi)
 0.00 :        135fee:       palignr $0xb,%xmm1,%xmm2
 0.00 :        135ff4:       movaps %xmm2,(%rdi)
 0.00 :        135ff7:       lea    0x80(%rdi),%rdi
 0.00 :        135ffe:       jae    135f70 <__memcpy_ssse3_back+0x1120>
 0.00 :        136004:       movdqu %xmm0,(%r8)
 0.00 :        136009:       add    $0x80,%rdx
 0.00 :        136010:       add    %rdx,%rdi
 0.00 :        136013:       add    %rdx,%rsi
```

```
 0.00 :        136016:       lea    0x3c413(%rip),%r11        # 172430 <null+0xbe0>
 0.00 :        13601d:       movslq (%r11,%rdx,4),%rdx
 0.00 :        136021:       lea    (%r11,%rdx,1),%rdx
 0.00 :        136025:       jmpq   *%rdx
 0.00 :        136027:       ud2
 0.00 :        136029:       nopl   0x0(%rax)
 0.00 :        136030:       movaps -0xb(%rsi),%xmm1
 0.00 :        136034:       movaps -0x1b(%rsi),%xmm2
 0.00 :        136038:       palignr $0xb,%xmm2,%xmm1
 0.00 :        13603e:       movaps %xmm1,-0x10(%rdi)
 0.00 :        136042:       movaps -0x2b(%rsi),%xmm3
 0.00 :        136046:       palignr $0xb,%xmm3,%xmm2
 0.00 :        13604c:       movaps %xmm2,-0x20(%rdi)
 0.00 :        136050:       movaps -0x3b(%rsi),%xmm4
 0.00 :        136054:       palignr $0xb,%xmm4,%xmm3
 0.00 :        13605a:       movaps %xmm3,-0x30(%rdi)
 0.00 :        13605e:       movaps -0x4b(%rsi),%xmm5
 0.00 :        136062:       palignr $0xb,%xmm5,%xmm4
 0.00 :        136068:       movaps %xmm4,-0x40(%rdi)
 0.00 :        13606c:       movaps -0x5b(%rsi),%xmm6
 0.00 :        136070:       palignr $0xb,%xmm6,%xmm5
 0.00 :        136076:       movaps %xmm5,-0x50(%rdi)
 0.00 :        13607a:       movaps -0x6b(%rsi),%xmm7
 0.00 :        13607e:       palignr $0xb,%xmm7,%xmm6
 0.00 :        136084:       movaps %xmm6,-0x60(%rdi)
 0.00 :        136088:       movaps -0x7b(%rsi),%xmm8
 0.00 :        13608d:       palignr $0xb,%xmm8,%xmm7
 0.00 :        136094:       movaps %xmm7,-0x70(%rdi)
 0.00 :        136098:       movaps -0x8b(%rsi),%xmm9
 0.00 :        1360a0:       palignr $0xb,%xmm9,%xmm8
 0.00 :        1360a7:       movaps %xmm8,-0x80(%rdi)
 0.00 :        1360ac:       sub    $0x80,%rdx
 0.00 :        1360b3:       lea    -0x80(%rdi),%rdi
 0.00 :        1360b7:       lea    -0x80(%rsi),%rsi
 0.00 :        1360bb:       jae    136030 <__memcpy_ssse3_back+0x11e0>
 0.00 :        1360c1:       movdqu %xmm0,(%r8)
 0.00 :        1360c6:       add    $0x80,%rdx
 0.00 :        1360cd:       sub    %rdx,%rdi
 0.00 :        1360d0:       sub    %rdx,%rsi
 0.00 :        1360d3:       lea    0x3c116(%rip),%r11        # 1721f0 <null+0x9a0>
 0.00 :        1360da:       movslq (%r11,%rdx,4),%rdx
 0.00 :        1360de:       lea    (%r11,%rdx,1),%rdx
 0.00 :        1360e2:       jmpq   *%rdx
 0.00 :        1360e4:       ud2
 0.00 :        1360e6:       nopw   %cs:0x0(%rax,%rax,1)
 0.00 :        1360f0:       sub    $0x80,%rdx
 0.00 :        1360f7:       movdqa -0xc(%rsi),%xmm1
 0.00 :        1360fc:       movaps 0x4(%rsi),%xmm2
 0.00 :        136100:       movaps 0x14(%rsi),%xmm3
 0.00 :        136104:       movaps 0x24(%rsi),%xmm4
 0.00 :        136108:       movaps 0x34(%rsi),%xmm5
 0.00 :        13610c:       movaps 0x44(%rsi),%xmm6
 0.00 :        136110:       movaps 0x54(%rsi),%xmm7
 0.00 :        136114:       movaps 0x64(%rsi),%xmm8
 0.00 :        136119:       movaps 0x74(%rsi),%xmm9
 0.00 :        13611e:       lea    0x80(%rsi),%rsi
 0.00 :        136125:       palignr $0xc,%xmm8,%xmm9
 0.00 :        13612c:       movaps %xmm9,0x70(%rdi)
 0.00 :        136131:       palignr $0xc,%xmm7,%xmm8
 0.00 :        136138:       movaps %xmm8,0x60(%rdi)
 0.00 :        13613d:       palignr $0xc,%xmm6,%xmm7
 0.00 :        136143:       movaps %xmm7,0x50(%rdi)
 0.00 :        136147:       palignr $0xc,%xmm5,%xmm6
 0.00 :        13614d:       movaps %xmm6,0x40(%rdi)
```

```
 0.00 :         136151:       palignr $0xc,%xmm4,%xmm5
 0.00 :         136157:       movaps %xmm5,0x30(%rdi)
 0.00 :         13615b:       palignr $0xc,%xmm3,%xmm4
 0.00 :         136161:       movaps %xmm4,0x20(%rdi)
 0.00 :         136165:       palignr $0xc,%xmm2,%xmm3
 0.00 :         13616b:       movaps %xmm3,0x10(%rdi)
 0.00 :         13616f:       palignr $0xc,%xmm1,%xmm2
 0.00 :         136175:       movaps %xmm2,(%rdi)
 0.00 :         136178:       lea    0x80(%rdi),%rdi
 0.00 :         13617f:       jae    1360f0 <__memcpy_ssse3_back+0x12a0>
 0.00 :         136185:       movdqu %xmm0,(%r8)
 0.00 :         13618a:       add    $0x80,%rdx
 0.00 :         136191:       add    %rdx,%rdi
 0.00 :         136194:       add    %rdx,%rsi
 0.00 :         136197:       lea    0x3c292(%rip),%r11        # 172430 <null+0xbe0>
 0.00 :         13619e:       movslq (%r11,%rdx,4),%rdx
 0.00 :         1361a2:       lea    (%r11,%rdx,1),%rdx
 0.00 :         1361a6:       jmpq   *%rdx
 0.00 :         1361a8:       ud2
 0.00 :         1361aa:       nopw   0x0(%rax,%rax,1)
 0.00 :         1361b0:       movaps -0xc(%rsi),%xmm1
 0.00 :         1361b4:       movaps -0x1c(%rsi),%xmm2
 0.00 :         1361b8:       palignr $0xc,%xmm2,%xmm1
 0.00 :         1361be:       movaps %xmm1,-0x10(%rdi)
 0.00 :         1361c2:       movaps -0x2c(%rsi),%xmm3
 0.00 :         1361c6:       palignr $0xc,%xmm3,%xmm2
 0.00 :         1361cc:       movaps %xmm2,-0x20(%rdi)
 0.00 :         1361d0:       movaps -0x3c(%rsi),%xmm4
 0.00 :         1361d4:       palignr $0xc,%xmm4,%xmm3
 0.00 :         1361da:       movaps %xmm3,-0x30(%rdi)
 0.00 :         1361de:       movaps -0x4c(%rsi),%xmm5
 0.00 :         1361e2:       palignr $0xc,%xmm5,%xmm4
 0.00 :         1361e8:       movaps %xmm4,-0x40(%rdi)
 0.00 :         1361ec:       movaps -0x5c(%rsi),%xmm6
 0.00 :         1361f0:       palignr $0xc,%xmm6,%xmm5
 0.00 :         1361f6:       movaps %xmm5,-0x50(%rdi)
 0.00 :         1361fa:       movaps -0x6c(%rsi),%xmm7
 0.00 :         1361fe:       palignr $0xc,%xmm7,%xmm6
 0.00 :         136204:       movaps %xmm6,-0x60(%rdi)
 0.00 :         136208:       movaps -0x7c(%rsi),%xmm8
 0.00 :         13620d:       palignr $0xc,%xmm8,%xmm7
 0.00 :         136214:       movaps %xmm7,-0x70(%rdi)
 0.00 :         136218:       movaps -0x8c(%rsi),%xmm9
 0.00 :         136220:       palignr $0xc,%xmm9,%xmm8
 0.00 :         136227:       movaps %xmm8,-0x80(%rdi)
 0.00 :         13622c:       sub    $0x80,%rdx
 0.00 :         136233:       lea    -0x80(%rdi),%rdi
 0.00 :         136237:       lea    -0x80(%rsi),%rsi
 0.00 :         13623b:       jae    1361b0 <__memcpy_ssse3_back+0x1360>
 0.00 :         136241:       movdqu %xmm0,(%r8)
 0.00 :         136246:       add    $0x80,%rdx
 0.00 :         13624d:       sub    %rdx,%rdi
 0.00 :         136250:       sub    %rdx,%rsi
 0.00 :         136253:       lea    0x3bf96(%rip),%r11        # 1721f0 <null+0x9a0>
 0.00 :         13625a:       movslq (%r11,%rdx,4),%rdx
 0.00 :         13625e:       lea    (%r11,%rdx,1),%rdx
 0.00 :         136262:       jmpq   *%rdx
 0.00 :         136264:       ud2
 0.00 :         136266:       nopw   %cs:0x0(%rax,%rax,1)
 0.00 :         136270:       sub    $0x80,%rdx
 0.00 :         136277:       movaps -0xd(%rsi),%xmm1
 0.00 :         13627b:       movaps 0x3(%rsi),%xmm2
 0.00 :         13627f:       movaps 0x13(%rsi),%xmm3
 0.00 :         136283:       movaps 0x23(%rsi),%xmm4
```

```
 0.00 :       136287:       movaps 0x33(%rsi),%xmm5
 0.00 :       13628b:       movaps 0x43(%rsi),%xmm6
 0.00 :       13628f:       movaps 0x53(%rsi),%xmm7
 0.00 :       136293:       movaps 0x63(%rsi),%xmm8
 0.00 :       136298:       movaps 0x73(%rsi),%xmm9
 0.00 :       13629d:       lea    0x80(%rsi),%rsi
 0.00 :       1362a4:       palignr $0xd,%xmm8,%xmm9
 0.00 :       1362ab:       movaps %xmm9,0x70(%rdi)
 0.00 :       1362b0:       palignr $0xd,%xmm7,%xmm8
 0.00 :       1362b7:       movaps %xmm8,0x60(%rdi)
 0.00 :       1362bc:       palignr $0xd,%xmm6,%xmm7
 0.00 :       1362c2:       movaps %xmm7,0x50(%rdi)
 0.00 :       1362c6:       palignr $0xd,%xmm5,%xmm6
 0.00 :       1362cc:       movaps %xmm6,0x40(%rdi)
 0.00 :       1362d0:       palignr $0xd,%xmm4,%xmm5
 0.00 :       1362d6:       movaps %xmm5,0x30(%rdi)
 0.00 :       1362da:       palignr $0xd,%xmm3,%xmm4
 0.00 :       1362e0:       movaps %xmm4,0x20(%rdi)
 0.00 :       1362e4:       palignr $0xd,%xmm2,%xmm3
 0.00 :       1362ea:       movaps %xmm3,0x10(%rdi)
 0.00 :       1362ee:       palignr $0xd,%xmm1,%xmm2
 0.00 :       1362f4:       movaps %xmm2,(%rdi)
 0.00 :       1362f7:       lea    0x80(%rdi),%rdi
 0.00 :       1362fe:       jae    136270 <__memcpy_ssse3_back+0x1420>
 0.00 :       136304:       movdqu %xmm0,(%r8)
 0.00 :       136309:       add    $0x80,%rdx
 0.00 :       136310:       add    %rdx,%rdi
 0.00 :       136313:       add    %rdx,%rsi
 0.00 :       136316:       lea    0x3c113(%rip),%r11        # 172430 <null+0xbe0>
 1.41 :       13631d:       movslq (%r11,%rdx,4),%rdx
 1.41 :       136321:       lea    (%r11,%rdx,1),%rdx
 0.00 :       136325:       jmpq   *%rdx
 0.00 :       136327:       ud2
 0.00 :       136329:       nopl   0x0(%rax)
 0.00 :       136330:       movaps -0xd(%rsi),%xmm1
 0.00 :       136334:       movaps -0x1d(%rsi),%xmm2
 0.00 :       136338:       palignr $0xd,%xmm2,%xmm1
 0.00 :       13633e:       movaps %xmm1,-0x10(%rdi)
 0.00 :       136342:       movaps -0x2d(%rsi),%xmm3
 0.00 :       136346:       palignr $0xd,%xmm3,%xmm2
 0.00 :       13634c:       movaps %xmm2,-0x20(%rdi)
 0.00 :       136350:       movaps -0x3d(%rsi),%xmm4
 0.00 :       136354:       palignr $0xd,%xmm4,%xmm3
 0.00 :       13635a:       movaps %xmm3,-0x30(%rdi)
 0.00 :       13635e:       movaps -0x4d(%rsi),%xmm5
 0.00 :       136362:       palignr $0xd,%xmm5,%xmm4
 0.00 :       136368:       movaps %xmm4,-0x40(%rdi)
 0.00 :       13636c:       movaps -0x5d(%rsi),%xmm6
 0.00 :       136370:       palignr $0xd,%xmm6,%xmm5
 0.00 :       136376:       movaps %xmm5,-0x50(%rdi)
 0.00 :       13637a:       movaps -0x6d(%rsi),%xmm7
 1.41 :       13637e:       palignr $0xd,%xmm7,%xmm6
 0.00 :       136384:       movaps %xmm6,-0x60(%rdi)
 0.00 :       136388:       movaps -0x7d(%rsi),%xmm8
 0.00 :       13638d:       palignr $0xd,%xmm8,%xmm7
 0.00 :       136394:       movaps %xmm7,-0x70(%rdi)
 0.00 :       136398:       movaps -0x8d(%rsi),%xmm9
 0.00 :       1363a0:       palignr $0xd,%xmm9,%xmm8
 0.00 :       1363a7:       movaps %xmm8,-0x80(%rdi)
 0.00 :       1363ac:       sub    $0x80,%rdx
 0.00 :       1363b3:       lea    -0x80(%rdi),%rdi
 0.00 :       1363b7:       lea    -0x80(%rsi),%rsi
 0.00 :       1363bb:       jae    136330 <__memcpy_ssse3_back+0x14e0>
 0.00 :       1363c1:       movdqu %xmm0,(%r8)
```

```
 0.00 :        1363c6:       add    $0x80,%rdx
 0.00 :        1363cd:       sub    %rdx,%rdi
 0.00 :        1363d0:       sub    %rdx,%rsi
 0.00 :        1363d3:       lea    0x3be16(%rip),%r11        # 1721f0 <null+0x9a0>
 0.00 :        1363da:       movslq (%r11,%rdx,4),%rdx
 0.00 :        1363de:       lea    (%r11,%rdx,1),%rdx
 0.00 :        1363e2:       jmpq   *%rdx
 0.00 :        1363e4:       ud2
 0.00 :        1363e6:       nopw   %cs:0x0(%rax,%rax,1)
 0.00 :        1363f0:       sub    $0x80,%rdx
 0.00 :        1363f7:       movaps -0xe(%rsi),%xmm1
 0.00 :        1363fb:       movaps 0x2(%rsi),%xmm2
 0.00 :        1363ff:       movaps 0x12(%rsi),%xmm3
 0.00 :        136403:       movaps 0x22(%rsi),%xmm4
 0.00 :        136407:       movaps 0x32(%rsi),%xmm5
 0.00 :        13640b:       movaps 0x42(%rsi),%xmm6
 0.00 :        13640f:       movaps 0x52(%rsi),%xmm7
 0.00 :        136413:       movaps 0x62(%rsi),%xmm8
 0.00 :        136418:       movaps 0x72(%rsi),%xmm9
 0.00 :        13641d:       lea    0x80(%rsi),%rsi
 0.00 :        136424:       palignr $0xe,%xmm8,%xmm9
 0.00 :        13642b:       movaps %xmm9,0x70(%rdi)
 0.00 :        136430:       palignr $0xe,%xmm7,%xmm8
 0.00 :        136437:       movaps %xmm8,0x60(%rdi)
 0.00 :        13643c:       palignr $0xe,%xmm6,%xmm7
 0.00 :        136442:       movaps %xmm7,0x50(%rdi)
 0.00 :        136446:       palignr $0xe,%xmm5,%xmm6
 0.00 :        13644c:       movaps %xmm6,0x40(%rdi)
 0.00 :        136450:       palignr $0xe,%xmm4,%xmm5
 0.00 :        136456:       movaps %xmm5,0x30(%rdi)
 0.00 :        13645a:       palignr $0xe,%xmm3,%xmm4
 0.00 :        136460:       movaps %xmm4,0x20(%rdi)
 0.00 :        136464:       palignr $0xe,%xmm2,%xmm3
 0.00 :        13646a:       movaps %xmm3,0x10(%rdi)
 0.00 :        13646e:       palignr $0xe,%xmm1,%xmm2
 0.00 :        136474:       movaps %xmm2,(%rdi)
 0.00 :        136477:       lea    0x80(%rdi),%rdi
 0.00 :        13647e:       jae    1363f0 <__memcpy_ssse3_back+0x15a0>
 0.00 :        136484:       movdqu %xmm0,(%r8)
 0.00 :        136489:       add    $0x80,%rdx
 0.00 :        136490:       add    %rdx,%rdi
 0.00 :        136493:       add    %rdx,%rsi
 0.00 :        136496:       lea    0x3bf93(%rip),%r11        # 172430 <null+0xbe0>
 0.00 :        13649d:       movslq (%r11,%rdx,4),%rdx
 0.00 :        1364a1:       lea    (%r11,%rdx,1),%rdx
 0.00 :        1364a5:       jmpq   *%rdx
 0.00 :        1364a7:       ud2
 0.00 :        1364a9:       nopl   0x0(%rax)
 0.00 :        1364b0:       movaps -0xe(%rsi),%xmm1
 0.00 :        1364b4:       movaps -0x1e(%rsi),%xmm2
 0.00 :        1364b8:       palignr $0xe,%xmm2,%xmm1
 0.00 :        1364be:       movaps %xmm1,-0x10(%rdi)
 0.00 :        1364c2:       movaps -0x2e(%rsi),%xmm3
 0.00 :        1364c6:       palignr $0xe,%xmm3,%xmm2
 0.00 :        1364cc:       movaps %xmm2,-0x20(%rdi)
 0.00 :        1364d0:       movaps -0x3e(%rsi),%xmm4
 0.00 :        1364d4:       palignr $0xe,%xmm4,%xmm3
 0.00 :        1364da:       movaps %xmm3,-0x30(%rdi)
 0.00 :        1364de:       movaps -0x4e(%rsi),%xmm5
 0.00 :        1364e2:       palignr $0xe,%xmm5,%xmm4
 0.00 :        1364e8:       movaps %xmm4,-0x40(%rdi)
 0.00 :        1364ec:       movaps -0x5e(%rsi),%xmm6
 0.00 :        1364f0:       palignr $0xe,%xmm6,%xmm5
 0.00 :        1364f6:       movaps %xmm5,-0x50(%rdi)
```

```
 0.00 :         1364fa:       movaps -0x6e(%rsi),%xmm7
 0.00 :         1364fe:       palignr $0xe,%xmm7,%xmm6
 0.00 :         136504:       movaps %xmm6,-0x60(%rdi)
 0.00 :         136508:       movaps -0x7e(%rsi),%xmm8
 0.00 :         13650d:       palignr $0xe,%xmm8,%xmm7
 0.00 :         136514:       movaps %xmm7,-0x70(%rdi)
 0.00 :         136518:       movaps -0x8e(%rsi),%xmm9
 0.00 :         136520:       palignr $0xe,%xmm9,%xmm8
 0.00 :         136527:       movaps %xmm8,-0x80(%rdi)
 0.00 :         13652c:       sub    $0x80,%rdx
 0.00 :         136533:       lea    -0x80(%rdi),%rdi
 0.00 :         136537:       lea    -0x80(%rsi),%rsi
 0.00 :         13653b:       jae    1364b0 <__memcpy_ssse3_back+0x1660>
 0.00 :         136541:       movdqu %xmm0,(%r8)
 0.00 :         136546:       add    $0x80,%rdx
 0.00 :         13654d:       sub    %rdx,%rdi
 0.00 :         136550:       sub    %rdx,%rsi
 0.00 :         136553:       lea    0x3bc96(%rip),%r11        # 1721f0 <null+0x9a0>
 0.00 :         13655a:       movslq (%r11,%rdx,4),%rdx
 0.00 :         13655e:       lea    (%r11,%rdx,1),%rdx
 0.00 :         136562:       jmpq   *%rdx
 0.00 :         136564:       ud2
 0.00 :         136566:       nopw   %cs:0x0(%rax,%rax,1)
 0.00 :         136570:       sub    $0x80,%rdx
 1.41 :         136577:       movaps -0xf(%rsi),%xmm1
 0.00 :         13657b:       movaps 0x1(%rsi),%xmm2
 0.00 :         13657f:       movaps 0x11(%rsi),%xmm3
 0.00 :         136583:       movaps 0x21(%rsi),%xmm4
 0.00 :         136587:       movaps 0x31(%rsi),%xmm5
 0.00 :         13658b:       movaps 0x41(%rsi),%xmm6
 0.00 :         13658f:       movaps 0x51(%rsi),%xmm7
 1.41 :         136593:       movaps 0x61(%rsi),%xmm8
 0.00 :         136598:       movaps 0x71(%rsi),%xmm9
 0.00 :         13659d:       lea    0x80(%rsi),%rsi
 0.00 :         1365a4:       palignr $0xf,%xmm8,%xmm9
 0.00 :         1365ab:       movaps %xmm9,0x70(%rdi)
 0.00 :         1365b0:       palignr $0xf,%xmm7,%xmm8
 0.00 :         1365b7:       movaps %xmm8,0x60(%rdi)
 0.00 :         1365bc:       palignr $0xf,%xmm6,%xmm7
 0.00 :         1365c2:       movaps %xmm7,0x50(%rdi)
 0.00 :         1365c6:       palignr $0xf,%xmm5,%xmm6
 0.00 :         1365cc:       movaps %xmm6,0x40(%rdi)
 0.00 :         1365d0:       palignr $0xf,%xmm4,%xmm5
 0.00 :         1365d6:       movaps %xmm5,0x30(%rdi)
 0.00 :         1365da:       palignr $0xf,%xmm3,%xmm4
 0.00 :         1365e0:       movaps %xmm4,0x20(%rdi)
 0.00 :         1365e4:       palignr $0xf,%xmm2,%xmm3
 0.00 :         1365ea:       movaps %xmm3,0x10(%rdi)
 0.00 :         1365ee:       palignr $0xf,%xmm1,%xmm2
 0.00 :         1365f4:       movaps %xmm2,(%rdi)
 0.00 :         1365f7:       lea    0x80(%rdi),%rdi
 0.00 :         1365fe:       jae    136570 <__memcpy_ssse3_back+0x1720>
 0.00 :         136604:       movdqu %xmm0,(%r8)
 0.00 :         136609:       add    $0x80,%rdx
 0.00 :         136610:       add    %rdx,%rdi
 0.00 :         136613:       add    %rdx,%rsi
 0.00 :         136616:       lea    0x3be13(%rip),%r11        # 172430 <null+0xbe0>
 0.00 :         13661d:       movslq (%r11,%rdx,4),%rdx
 0.00 :         136621:       lea    (%r11,%rdx,1),%rdx
 0.00 :         136625:       jmpq   *%rdx
 0.00 :         136627:       ud2
 0.00 :         136629:       nopl   0x0(%rax)
 0.00 :         136630:       movaps -0xf(%rsi),%xmm1
 1.41 :         136634:       movaps -0x1f(%rsi),%xmm2
```

```
 14.08 :        136638:       palignr $0xf,%xmm2,%xmm1
  2.82 :        13663e:       movaps %xmm1,-0x10(%rdi)
  0.00 :        136642:       movaps -0x2f(%rsi),%xmm3
  0.00 :        136646:       palignr $0xf,%xmm3,%xmm2
  0.00 :        13664c:       movaps %xmm2,-0x20(%rdi)
  1.41 :        136650:       movaps -0x3f(%rsi),%xmm4
  0.00 :        136654:       palignr $0xf,%xmm4,%xmm3
  0.00 :        13665a:       movaps %xmm3,-0x30(%rdi)
  1.41 :        13665e:       movaps -0x4f(%rsi),%xmm5
  0.00 :        136662:       palignr $0xf,%xmm5,%xmm4
  1.41 :        136668:       movaps %xmm4,-0x40(%rdi)
  1.41 :        13666c:       movaps -0x5f(%rsi),%xmm6
 11.27 :        136670:       palignr $0xf,%xmm6,%xmm5
  0.00 :        136676:       movaps %xmm5,-0x50(%rdi)
  1.41 :        13667a:       movaps -0x6f(%rsi),%xmm7
  0.00 :        13667e:       palignr $0xf,%xmm7,%xmm6
  0.00 :        136684:       movaps %xmm6,-0x60(%rdi)
  1.41 :        136688:       movaps -0x7f(%rsi),%xmm8
  0.00 :        13668d:       palignr $0xf,%xmm8,%xmm7
  0.00 :        136694:       movaps %xmm7,-0x70(%rdi)
  0.00 :        136698:       movaps -0x8f(%rsi),%xmm9
  0.00 :        1366a0:       palignr $0xf,%xmm9,%xmm8
  0.00 :        1366a7:       movaps %xmm8,-0x80(%rdi)
  0.00 :        1366ac:       sub    $0x80,%rdx
  1.41 :        1366b3:       lea    -0x80(%rdi),%rdi
  0.00 :        1366b7:       lea    -0x80(%rsi),%rsi
  1.41 :        1366bb:       jae    136630 <__memcpy_ssse3_back+0x17e0>
  0.00 :        1366c1:       movdqu %xmm0,(%r8)
  0.00 :        1366c6:       add    $0x80,%rdx
  0.00 :        1366cd:       sub    %rdx,%rdi
  0.00 :        1366d0:       sub    %rdx,%rsi
  0.00 :        1366d3:       lea    0x3bb16(%rip),%r11        # 1721f0 <null+0x9a0>
  0.00 :        1366da:       movslq (%r11,%rdx,4),%rdx
  0.00 :        1366de:       lea    (%r11,%rdx,1),%rdx
  0.00 :        1366e2:       jmpq   *%rdx
  0.00 :        1366e4:       ud2
  0.00 :        1366e6:       nopw   %cs:0x0(%rax,%rax,1)
  0.00 :        1366f0:       movdqu (%rsi),%xmm1
  0.00 :        1366f4:       movdqu %xmm0,(%r8)
  0.00 :        1366f9:       movdqa %xmm1,(%rdi)
  0.00 :        1366fd:       sub    $0x10,%rdx
  0.00 :        136701:       add    $0x10,%rsi
  0.00 :        136705:       add    $0x10,%rdi
  0.00 :        136709:       mov    0x270ad8(%rip),%rcx        # 3a71e8 <__x86_64_shared_cache_size_half>
  0.00 :        136710:       cmp    %rcx,%rdx
  0.00 :        136713:       ja     136718 <__memcpy_ssse3_back+0x18c8>
  0.00 :        136715:       mov    %rdx,%rcx
  0.00 :        136718:       sub    %rcx,%rdx
  0.00 :        13671b:       cmp    $0x1000,%rdx
  0.00 :        136722:       jbe    1367ce <__memcpy_ssse3_back+0x197e>
  0.00 :        136728:       mov    %rcx,%r9
  0.00 :        13672b:       shl    $0x3,%r9
  0.00 :        13672f:       cmp    %r9,%rdx
  0.00 :        136732:       jbe    13673a <__memcpy_ssse3_back+0x18ea>
  0.00 :        136734:       add    %rcx,%rdx
  0.00 :        136737:       xor    %rcx,%rcx
  0.00 :        13673a:       sub    $0x80,%rdx
  0.00 :        136741:       sub    $0x80,%rdx
  0.00 :        136748:       prefetcht0 0x200(%rsi)
  0.00 :        13674f:       prefetcht0 0x300(%rsi)
  0.00 :        136756:       movdqu (%rsi),%xmm0
  0.00 :        13675a:       movdqu 0x10(%rsi),%xmm1
  0.00 :        13675f:       movdqu 0x20(%rsi),%xmm2
  0.00 :        136764:       movdqu 0x30(%rsi),%xmm3
```

```
 0.00 :         136769:       movdqu 0x40(%rsi),%xmm4
 0.00 :         13676e:       movdqu 0x50(%rsi),%xmm5
 0.00 :         136773:       movdqu 0x60(%rsi),%xmm6
 0.00 :         136778:       movdqu 0x70(%rsi),%xmm7
 0.00 :         13677d:       lfence
 0.00 :         136780:       movntdq %xmm0,(%rdi)
 0.00 :         136784:       movntdq %xmm1,0x10(%rdi)
 0.00 :         136789:       movntdq %xmm2,0x20(%rdi)
 0.00 :         13678e:       movntdq %xmm3,0x30(%rdi)
 0.00 :         136793:       movntdq %xmm4,0x40(%rdi)
 0.00 :         136798:       movntdq %xmm5,0x50(%rdi)
 0.00 :         13679d:       movntdq %xmm6,0x60(%rdi)
 0.00 :         1367a2:       movntdq %xmm7,0x70(%rdi)
 0.00 :         1367a7:       lea    0x80(%rsi),%rsi
 0.00 :         1367ae:       lea    0x80(%rdi),%rdi
 0.00 :         1367b5:       jae    136741 <__memcpy_ssse3_back+0x18f1>
 0.00 :         1367b7:       sfence
 0.00 :         1367ba:       cmp    $0x80,%rcx
 0.00 :         1367c1:       jb     13685d <__memcpy_ssse3_back+0x1a0d>
 0.00 :         1367c7:       add    $0x80,%rdx
 0.00 :         1367ce:       add    %rcx,%rdx
 0.00 :         1367d1:       sub    $0x80,%rdx
 0.00 :         1367d8:       prefetchnta 0x1c0(%rsi)
 0.00 :         1367df:       prefetchnta 0x280(%rsi)
 0.00 :         1367e6:       prefetchnta 0x1c0(%rdi)
 0.00 :         1367ed:       prefetchnta 0x280(%rdi)
 0.00 :         1367f4:       sub    $0x80,%rdx
 0.00 :         1367fb:       movdqu (%rsi),%xmm0
 0.00 :         1367ff:       movdqu 0x10(%rsi),%xmm1
 0.00 :         136804:       movdqu 0x20(%rsi),%xmm2
 0.00 :         136809:       movdqu 0x30(%rsi),%xmm3
 0.00 :         13680e:       movdqu 0x40(%rsi),%xmm4
 0.00 :         136813:       movdqu 0x50(%rsi),%xmm5
 0.00 :         136818:       movdqu 0x60(%rsi),%xmm6
 0.00 :         13681d:       movdqu 0x70(%rsi),%xmm7
 0.00 :         136822:       movdqa %xmm0,(%rdi)
 0.00 :         136826:       movdqa %xmm1,0x10(%rdi)
 0.00 :         13682b:       movdqa %xmm2,0x20(%rdi)
 0.00 :         136830:       movdqa %xmm3,0x30(%rdi)
 0.00 :         136835:       movdqa %xmm4,0x40(%rdi)
 0.00 :         13683a:       movdqa %xmm5,0x50(%rdi)
 0.00 :         13683f:       movdqa %xmm6,0x60(%rdi)
 0.00 :         136844:       movdqa %xmm7,0x70(%rdi)
 0.00 :         136849:       lea    0x80(%rsi),%rsi
 0.00 :         136850:       lea    0x80(%rdi),%rdi
 0.00 :         136857:       jae    1367d8 <__memcpy_ssse3_back+0x1988>
 0.00 :         13685d:       add    $0x80,%rdx
 0.00 :         136864:       add    %rdx,%rsi
 0.00 :         136867:       add    %rdx,%rdi
 0.00 :         13686a:       lea    0x3bbbf(%rip),%r11        # 172430 <null+0xbe0>
 0.00 :         136871:       movslq (%r11,%rdx,4),%rdx
 0.00 :         136875:       lea    (%r11,%rdx,1),%rdx
 0.00 :         136879:       jmpq   *%rdx
 0.00 :         13687b:       ud2
 0.00 :         13687d:       nopl   (%rax)
 0.00 :         136880:       add    %rdx,%rsi
 0.00 :         136883:       add    %rdx,%rdi
 0.00 :         136886:       movdqu -0x10(%rsi),%xmm0
 0.00 :         13688b:       lea    -0x10(%rdi),%r8
 0.00 :         13688f:       mov    %rdi,%r9
 0.00 :         136892:       and    $0xfffffffffffffff0,%rdi
 0.00 :         136896:       sub    %rdi,%r9
 0.00 :         136899:       sub    %r9,%rsi
 0.00 :         13689c:       sub    %r9,%rdx
```

```
 0.00 :        13689f:       mov    0x270942(%rip),%rcx        # 3a71e8 <__x86_64_shared_cache_size_half>
 0.00 :        1368a6:       cmp    %rcx,%rdx
 0.00 :        1368a9:       ja     1368ae <__memcpy_ssse3_back+0x1a5e>
 0.00 :        1368ab:       mov    %rdx,%rcx
 0.00 :        1368ae:       sub    %rcx,%rdx
 0.00 :        1368b1:       cmp    $0x1000,%rdx
 0.00 :        1368b8:       jbe    136962 <__memcpy_ssse3_back+0x1b12>
 0.00 :        1368be:       mov    %rcx,%r9
 0.00 :        1368c1:       shl    $0x3,%r9
 0.00 :        1368c5:       cmp    %r9,%rdx
 0.00 :        1368c8:       jbe    1368d0 <__memcpy_ssse3_back+0x1a80>
 0.00 :        1368ca:       add    %rcx,%rdx
 0.00 :        1368cd:       xor    %rcx,%rcx
 0.00 :        1368d0:       sub    $0x80,%rdx
 0.00 :        1368d7:       sub    $0x80,%rdx
 0.00 :        1368de:       prefetcht0 -0x200(%rsi)
 0.00 :        1368e5:       prefetcht0 -0x300(%rsi)
 0.00 :        1368ec:       movdqu -0x10(%rsi),%xmm1
 0.00 :        1368f1:       movdqu -0x20(%rsi),%xmm2
 0.00 :        1368f6:       movdqu -0x30(%rsi),%xmm3
 0.00 :        1368fb:       movdqu -0x40(%rsi),%xmm4
 0.00 :        136900:       movdqu -0x50(%rsi),%xmm5
 0.00 :        136905:       movdqu -0x60(%rsi),%xmm6
 0.00 :        13690a:       movdqu -0x70(%rsi),%xmm7
 0.00 :        13690f:       movdqu -0x80(%rsi),%xmm8
 0.00 :        136915:       lfence
 0.00 :        136918:       movntdq %xmm1,-0x10(%rdi)
 0.00 :        13691d:       movntdq %xmm2,-0x20(%rdi)
 0.00 :        136922:       movntdq %xmm3,-0x30(%rdi)
 0.00 :        136927:       movntdq %xmm4,-0x40(%rdi)
 0.00 :        13692c:       movntdq %xmm5,-0x50(%rdi)
 0.00 :        136931:       movntdq %xmm6,-0x60(%rdi)
 0.00 :        136936:       movntdq %xmm7,-0x70(%rdi)
 0.00 :        13693b:       movntdq %xmm8,-0x80(%rdi)
 0.00 :        136941:       lea    -0x80(%rsi),%rsi
 0.00 :        136945:       lea    -0x80(%rdi),%rdi
 0.00 :        136949:       jae    1368d7 <__memcpy_ssse3_back+0x1a87>
 0.00 :        13694b:       sfence
 0.00 :        13694e:       cmp    $0x80,%rcx
 0.00 :        136955:       jb     1369eb <__memcpy_ssse3_back+0x1b9b>
 0.00 :        13695b:       add    $0x80,%rdx
 0.00 :        136962:       add    %rcx,%rdx
 0.00 :        136965:       sub    $0x80,%rdx
 0.00 :        13696c:       prefetchnta -0x1c0(%rsi)
 0.00 :        136973:       prefetchnta -0x280(%rsi)
 0.00 :        13697a:       prefetchnta -0x1c0(%rdi)
 0.00 :        136981:       prefetchnta -0x280(%rdi)
 0.00 :        136988:       sub    $0x80,%rdx
 0.00 :        13698f:       movdqu -0x10(%rsi),%xmm1
 0.00 :        136994:       movdqu -0x20(%rsi),%xmm2
 0.00 :        136999:       movdqu -0x30(%rsi),%xmm3
 0.00 :        13699e:       movdqu -0x40(%rsi),%xmm4
 0.00 :        1369a3:       movdqu -0x50(%rsi),%xmm5
 0.00 :        1369a8:       movdqu -0x60(%rsi),%xmm6
 0.00 :        1369ad:       movdqu -0x70(%rsi),%xmm7
 0.00 :        1369b2:       movdqu -0x80(%rsi),%xmm8
 0.00 :        1369b8:       movdqa %xmm1,-0x10(%rdi)
 0.00 :        1369bd:       movdqa %xmm2,-0x20(%rdi)
 0.00 :        1369c2:       movdqa %xmm3,-0x30(%rdi)
 0.00 :        1369c7:       movdqa %xmm4,-0x40(%rdi)
 0.00 :        1369cc:       movdqa %xmm5,-0x50(%rdi)
 0.00 :        1369d1:       movdqa %xmm6,-0x60(%rdi)
 0.00 :        1369d6:       movdqa %xmm7,-0x70(%rdi)
 0.00 :        1369db:       movdqa %xmm8,-0x80(%rdi)
```

```
 0.00 :         1369e1:       lea    -0x80(%rsi),%rsi
 0.00 :         1369e5:       lea    -0x80(%rdi),%rdi
 0.00 :         1369e9:       jae    13696c <__memcpy_ssse3_back+0x1b1c>
 0.00 :         1369eb:       movdqu %xmm0,(%r8)
 0.00 :         1369f0:       add    $0x80,%rdx
 0.00 :         1369f7:       sub    %rdx,%rsi
 0.00 :         1369fa:       sub    %rdx,%rdi
 0.00 :         1369fd:       lea    0x3b7ec(%rip),%r11        # 1721f0 <null+0x9a0>
 0.00 :         136a04:       movslq (%r11,%rdx,4),%rdx
 0.00 :         136a08:       lea    (%r11,%rdx,1),%rdx
 0.00 :         136a0c:       jmpq   *%rdx
 0.00 :         136a0e:       ud2
 0.00 :         136a10:       lddqu  -0x80(%rsi),%xmm0
 0.00 :         136a15:       movdqu %xmm0,-0x80(%rdi)
 0.00 :         136a1a:       lddqu  -0x70(%rsi),%xmm0
 0.00 :         136a1f:       movdqu %xmm0,-0x70(%rdi)
 0.00 :         136a24:       lddqu  -0x60(%rsi),%xmm0
 0.00 :         136a29:       movdqu %xmm0,-0x60(%rdi)
 0.00 :         136a2e:       lddqu  -0x50(%rsi),%xmm0
 0.00 :         136a33:       movdqu %xmm0,-0x50(%rdi)
 0.00 :         136a38:       lddqu  -0x40(%rsi),%xmm0
 0.00 :         136a3d:       movdqu %xmm0,-0x40(%rdi)
 0.00 :         136a42:       lddqu  -0x30(%rsi),%xmm0
 0.00 :         136a47:       movdqu %xmm0,-0x30(%rdi)
 0.00 :         136a4c:       lddqu  -0x20(%rsi),%xmm0
 0.00 :         136a51:       movdqu %xmm0,-0x20(%rdi)
 0.00 :         136a56:       lddqu  -0x10(%rsi),%xmm0
 0.00 :         136a5b:       movdqu %xmm0,-0x10(%rdi)
 0.00 :         136a60:       retq
 0.00 :         136a61:       data32 data32 data32 data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :         136a70:       lddqu  -0x8f(%rsi),%xmm0
 0.00 :         136a78:       movdqu %xmm0,-0x8f(%rdi)
 0.00 :         136a80:       lddqu  -0x7f(%rsi),%xmm0
 0.00 :         136a85:       movdqu %xmm0,-0x7f(%rdi)
 0.00 :         136a8a:       lddqu  -0x6f(%rsi),%xmm0
 0.00 :         136a8f:       movdqu %xmm0,-0x6f(%rdi)
 0.00 :         136a94:       lddqu  -0x5f(%rsi),%xmm0
 0.00 :         136a99:       movdqu %xmm0,-0x5f(%rdi)
 0.00 :         136a9e:       lddqu  -0x4f(%rsi),%xmm0
 0.00 :         136aa3:       movdqu %xmm0,-0x4f(%rdi)
 0.00 :         136aa8:       lddqu  -0x3f(%rsi),%xmm0
 0.00 :         136aad:       movdqu %xmm0,-0x3f(%rdi)
 0.00 :         136ab2:       lddqu  -0x2f(%rsi),%xmm0
 0.00 :         136ab7:       movdqu %xmm0,-0x2f(%rdi)
 0.00 :         136abc:       lddqu  -0x1f(%rsi),%xmm0
 0.00 :         136ac1:       lddqu  -0x10(%rsi),%xmm1
 0.00 :         136ac6:       movdqu %xmm0,-0x1f(%rdi)
 0.00 :         136acb:       movdqu %xmm1,-0x10(%rdi)
 0.00 :         136ad0:       retq
 0.00 :         136ad1:       data32 data32 data32 data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :         136ae0:       mov    -0xf(%rsi),%rdx
 0.00 :         136ae4:       mov    -0x8(%rsi),%rcx
 0.00 :         136ae8:       mov    %rdx,-0xf(%rdi)
 0.00 :         136aec:       mov    %rcx,-0x8(%rdi)
 0.00 :         136af0:       retq
 0.00 :         136af1:       data32 data32 data32 data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :         136b00:       lddqu  -0x8e(%rsi),%xmm0
 0.00 :         136b08:       movdqu %xmm0,-0x8e(%rdi)
 0.00 :         136b10:       lddqu  -0x7e(%rsi),%xmm0
 0.00 :         136b15:       movdqu %xmm0,-0x7e(%rdi)
 0.00 :         136b1a:       lddqu  -0x6e(%rsi),%xmm0
 0.00 :         136b1f:       movdqu %xmm0,-0x6e(%rdi)
 0.00 :         136b24:       lddqu  -0x5e(%rsi),%xmm0
 0.00 :         136b29:       movdqu %xmm0,-0x5e(%rdi)
```

```
 0.00 :        136b2e:       lddqu  -0x4e(%rsi),%xmm0
 0.00 :        136b33:       movdqu %xmm0,-0x4e(%rdi)
 0.00 :        136b38:       lddqu  -0x3e(%rsi),%xmm0
 0.00 :        136b3d:       movdqu %xmm0,-0x3e(%rdi)
 0.00 :        136b42:       lddqu  -0x2e(%rsi),%xmm0
 0.00 :        136b47:       movdqu %xmm0,-0x2e(%rdi)
 0.00 :        136b4c:       lddqu  -0x1e(%rsi),%xmm0
 0.00 :        136b51:       lddqu  -0x10(%rsi),%xmm1
 0.00 :        136b56:       movdqu %xmm0,-0x1e(%rdi)
 0.00 :        136b5b:       movdqu %xmm1,-0x10(%rdi)
 0.00 :        136b60:       retq
 0.00 :        136b61:       data32 data32 data32 data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :        136b70:       mov    -0xe(%rsi),%rdx
 0.00 :        136b74:       mov    -0x8(%rsi),%rcx
 0.00 :        136b78:       mov    %rdx,-0xe(%rdi)
 0.00 :        136b7c:       mov    %rcx,-0x8(%rdi)
 0.00 :        136b80:       retq
 0.00 :        136b81:       data32 data32 data32 data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :        136b90:       lddqu  -0x8d(%rsi),%xmm0
 0.00 :        136b98:       movdqu %xmm0,-0x8d(%rdi)
 0.00 :        136ba0:       lddqu  -0x7d(%rsi),%xmm0
 0.00 :        136ba5:       movdqu %xmm0,-0x7d(%rdi)
 0.00 :        136baa:       lddqu  -0x6d(%rsi),%xmm0
 0.00 :        136baf:       movdqu %xmm0,-0x6d(%rdi)
 0.00 :        136bb4:       lddqu  -0x5d(%rsi),%xmm0
 0.00 :        136bb9:       movdqu %xmm0,-0x5d(%rdi)
 0.00 :        136bbe:       lddqu  -0x4d(%rsi),%xmm0
 0.00 :        136bc3:       movdqu %xmm0,-0x4d(%rdi)
 0.00 :        136bc8:       lddqu  -0x3d(%rsi),%xmm0
 0.00 :        136bcd:       movdqu %xmm0,-0x3d(%rdi)
 0.00 :        136bd2:       lddqu  -0x2d(%rsi),%xmm0
 0.00 :        136bd7:       movdqu %xmm0,-0x2d(%rdi)
 0.00 :        136bdc:       lddqu  -0x1d(%rsi),%xmm0
 0.00 :        136be1:       lddqu  -0x10(%rsi),%xmm1
 0.00 :        136be6:       movdqu %xmm0,-0x1d(%rdi)
 0.00 :        136beb:       movdqu %xmm1,-0x10(%rdi)
 0.00 :        136bf0:       retq
 0.00 :        136bf1:       data32 data32 data32 data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :        136c00:       mov    -0xd(%rsi),%rdx
 0.00 :        136c04:       mov    -0x8(%rsi),%rcx
 0.00 :        136c08:       mov    %rdx,-0xd(%rdi)
 0.00 :        136c0c:       mov    %rcx,-0x8(%rdi)
 0.00 :        136c10:       retq
 0.00 :        136c11:       data32 data32 data32 data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :        136c20:       lddqu  -0x8c(%rsi),%xmm0
 0.00 :        136c28:       movdqu %xmm0,-0x8c(%rdi)
 0.00 :        136c30:       lddqu  -0x7c(%rsi),%xmm0
 0.00 :        136c35:       movdqu %xmm0,-0x7c(%rdi)
 0.00 :        136c3a:       lddqu  -0x6c(%rsi),%xmm0
 0.00 :        136c3f:       movdqu %xmm0,-0x6c(%rdi)
 0.00 :        136c44:       lddqu  -0x5c(%rsi),%xmm0
 0.00 :        136c49:       movdqu %xmm0,-0x5c(%rdi)
 0.00 :        136c4e:       lddqu  -0x4c(%rsi),%xmm0
 0.00 :        136c53:       movdqu %xmm0,-0x4c(%rdi)
 0.00 :        136c58:       lddqu  -0x3c(%rsi),%xmm0
 0.00 :        136c5d:       movdqu %xmm0,-0x3c(%rdi)
 0.00 :        136c62:       lddqu  -0x2c(%rsi),%xmm0
 0.00 :        136c67:       movdqu %xmm0,-0x2c(%rdi)
 0.00 :        136c6c:       lddqu  -0x1c(%rsi),%xmm0
 0.00 :        136c71:       lddqu  -0x10(%rsi),%xmm1
 0.00 :        136c76:       movdqu %xmm0,-0x1c(%rdi)
 0.00 :        136c7b:       movdqu %xmm1,-0x10(%rdi)
 0.00 :        136c80:       retq
 0.00 :        136c81:       data32 data32 data32 data32 nopw %cs:0x0(%rax,%rax,1)
```

```
 0.00 :        136c90:       mov    -0xc(%rsi),%rdx
 0.00 :        136c94:       mov    -0x4(%rsi),%ecx
 0.00 :        136c97:       mov    %rdx,-0xc(%rdi)
 0.00 :        136c9b:       mov    %ecx,-0x4(%rdi)
 0.00 :        136c9e:       retq
 0.00 :        136c9f:       nop
 0.00 :        136ca0:       lddqu  -0x8b(%rsi),%xmm0
 0.00 :        136ca8:       movdqu %xmm0,-0x8b(%rdi)
 0.00 :        136cb0:       lddqu  -0x7b(%rsi),%xmm0
 0.00 :        136cb5:       movdqu %xmm0,-0x7b(%rdi)
 0.00 :        136cba:       lddqu  -0x6b(%rsi),%xmm0
 0.00 :        136cbf:       movdqu %xmm0,-0x6b(%rdi)
 0.00 :        136cc4:       lddqu  -0x5b(%rsi),%xmm0
 0.00 :        136cc9:       movdqu %xmm0,-0x5b(%rdi)
 0.00 :        136cce:       lddqu  -0x4b(%rsi),%xmm0
 0.00 :        136cd3:       movdqu %xmm0,-0x4b(%rdi)
 0.00 :        136cd8:       lddqu  -0x3b(%rsi),%xmm0
 0.00 :        136cdd:       movdqu %xmm0,-0x3b(%rdi)
 0.00 :        136ce2:       lddqu  -0x2b(%rsi),%xmm0
 0.00 :        136ce7:       movdqu %xmm0,-0x2b(%rdi)
 0.00 :        136cec:       lddqu  -0x1b(%rsi),%xmm0
 0.00 :        136cf1:       lddqu  -0x10(%rsi),%xmm1
 0.00 :        136cf6:       movdqu %xmm0,-0x1b(%rdi)
 0.00 :        136cfb:       movdqu %xmm1,-0x10(%rdi)
 0.00 :        136d00:       retq
 0.00 :        136d01:       data32 data32 data32 data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :        136d10:       mov    -0xb(%rsi),%rdx
 0.00 :        136d14:       mov    -0x4(%rsi),%ecx
 0.00 :        136d17:       mov    %rdx,-0xb(%rdi)
 0.00 :        136d1b:       mov    %ecx,-0x4(%rdi)
 0.00 :        136d1e:       retq
 0.00 :        136d1f:       nop
 0.00 :        136d20:       lddqu  -0x8a(%rsi),%xmm0
 0.00 :        136d28:       movdqu %xmm0,-0x8a(%rdi)
 0.00 :        136d30:       lddqu  -0x7a(%rsi),%xmm0
 0.00 :        136d35:       movdqu %xmm0,-0x7a(%rdi)
 0.00 :        136d3a:       lddqu  -0x6a(%rsi),%xmm0
 0.00 :        136d3f:       movdqu %xmm0,-0x6a(%rdi)
 0.00 :        136d44:       lddqu  -0x5a(%rsi),%xmm0
 0.00 :        136d49:       movdqu %xmm0,-0x5a(%rdi)
 0.00 :        136d4e:       lddqu  -0x4a(%rsi),%xmm0
 0.00 :        136d53:       movdqu %xmm0,-0x4a(%rdi)
 0.00 :        136d58:       lddqu  -0x3a(%rsi),%xmm0
 0.00 :        136d5d:       movdqu %xmm0,-0x3a(%rdi)
 0.00 :        136d62:       lddqu  -0x2a(%rsi),%xmm0
 0.00 :        136d67:       movdqu %xmm0,-0x2a(%rdi)
 0.00 :        136d6c:       lddqu  -0x1a(%rsi),%xmm0
 0.00 :        136d71:       lddqu  -0x10(%rsi),%xmm1
 0.00 :        136d76:       movdqu %xmm0,-0x1a(%rdi)
 0.00 :        136d7b:       movdqu %xmm1,-0x10(%rdi)
 0.00 :        136d80:       retq
 0.00 :        136d81:       data32 data32 data32 data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :        136d90:       mov    -0xa(%rsi),%rdx
 0.00 :        136d94:       mov    -0x4(%rsi),%ecx
 0.00 :        136d97:       mov    %rdx,-0xa(%rdi)
 0.00 :        136d9b:       mov    %ecx,-0x4(%rdi)
 0.00 :        136d9e:       retq
 0.00 :        136d9f:       nop
 0.00 :        136da0:       lddqu  -0x89(%rsi),%xmm0
 0.00 :        136da8:       movdqu %xmm0,-0x89(%rdi)
 0.00 :        136db0:       lddqu  -0x79(%rsi),%xmm0
 0.00 :        136db5:       movdqu %xmm0,-0x79(%rdi)
 0.00 :        136dba:       lddqu  -0x69(%rsi),%xmm0
 0.00 :        136dbf:       movdqu %xmm0,-0x69(%rdi)
```

```
 0.00 :         136dc4:       lddqu  -0x59(%rsi),%xmm0
 0.00 :         136dc9:       movdqu %xmm0,-0x59(%rdi)
 0.00 :         136dce:       lddqu  -0x49(%rsi),%xmm0
 0.00 :         136dd3:       movdqu %xmm0,-0x49(%rdi)
 0.00 :         136dd8:       lddqu  -0x39(%rsi),%xmm0
 0.00 :         136ddd:       movdqu %xmm0,-0x39(%rdi)
 0.00 :         136de2:       lddqu  -0x29(%rsi),%xmm0
 0.00 :         136de7:       movdqu %xmm0,-0x29(%rdi)
 0.00 :         136dec:       lddqu  -0x19(%rsi),%xmm0
 0.00 :         136df1:       lddqu  -0x10(%rsi),%xmm1
 0.00 :         136df6:       movdqu %xmm0,-0x19(%rdi)
 0.00 :         136dfb:       movdqu %xmm1,-0x10(%rdi)
 0.00 :         136e00:       retq
 0.00 :         136e01:       data32 data32 data32 data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :         136e10:       mov    -0x9(%rsi),%rdx
 0.00 :         136e14:       mov    -0x4(%rsi),%ecx
 0.00 :         136e17:       mov    %rdx,-0x9(%rdi)
 0.00 :         136e1b:       mov    %ecx,-0x4(%rdi)
 0.00 :         136e1e:       retq
 0.00 :         136e1f:       nop
 0.00 :         136e20:       lddqu  -0x88(%rsi),%xmm0
 0.00 :         136e28:       movdqu %xmm0,-0x88(%rdi)
 0.00 :         136e30:       lddqu  -0x78(%rsi),%xmm0
 0.00 :         136e35:       movdqu %xmm0,-0x78(%rdi)
 0.00 :         136e3a:       lddqu  -0x68(%rsi),%xmm0
 0.00 :         136e3f:       movdqu %xmm0,-0x68(%rdi)
 0.00 :         136e44:       lddqu  -0x58(%rsi),%xmm0
 0.00 :         136e49:       movdqu %xmm0,-0x58(%rdi)
 0.00 :         136e4e:       lddqu  -0x48(%rsi),%xmm0
 0.00 :         136e53:       movdqu %xmm0,-0x48(%rdi)
 0.00 :         136e58:       lddqu  -0x38(%rsi),%xmm0
 0.00 :         136e5d:       movdqu %xmm0,-0x38(%rdi)
 0.00 :         136e62:       lddqu  -0x28(%rsi),%xmm0
 0.00 :         136e67:       movdqu %xmm0,-0x28(%rdi)
 0.00 :         136e6c:       lddqu  -0x18(%rsi),%xmm0
 0.00 :         136e71:       lddqu  -0x10(%rsi),%xmm1
 0.00 :         136e76:       movdqu %xmm0,-0x18(%rdi)
 0.00 :         136e7b:       movdqu %xmm1,-0x10(%rdi)
 0.00 :         136e80:       retq
 0.00 :         136e81:       data32 data32 data32 data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :         136e90:       mov    -0x8(%rsi),%rdx
 0.00 :         136e94:       mov    %rdx,-0x8(%rdi)
 0.00 :         136e98:       retq
 0.00 :         136e99:       nopl   0x0(%rax)
 0.00 :         136ea0:       lddqu  -0x87(%rsi),%xmm0
 0.00 :         136ea8:       movdqu %xmm0,-0x87(%rdi)
 0.00 :         136eb0:       lddqu  -0x77(%rsi),%xmm0
 0.00 :         136eb5:       movdqu %xmm0,-0x77(%rdi)
 0.00 :         136eba:       lddqu  -0x67(%rsi),%xmm0
 0.00 :         136ebf:       movdqu %xmm0,-0x67(%rdi)
 0.00 :         136ec4:       lddqu  -0x57(%rsi),%xmm0
 0.00 :         136ec9:       movdqu %xmm0,-0x57(%rdi)
 0.00 :         136ece:       lddqu  -0x47(%rsi),%xmm0
 0.00 :         136ed3:       movdqu %xmm0,-0x47(%rdi)
 0.00 :         136ed8:       lddqu  -0x37(%rsi),%xmm0
 0.00 :         136edd:       movdqu %xmm0,-0x37(%rdi)
 0.00 :         136ee2:       lddqu  -0x27(%rsi),%xmm0
 0.00 :         136ee7:       movdqu %xmm0,-0x27(%rdi)
 0.00 :         136eec:       lddqu  -0x17(%rsi),%xmm0
 0.00 :         136ef1:       lddqu  -0x10(%rsi),%xmm1
 0.00 :         136ef6:       movdqu %xmm0,-0x17(%rdi)
 0.00 :         136efb:       movdqu %xmm1,-0x10(%rdi)
 0.00 :         136f00:       retq
 0.00 :         136f01:       data32 data32 data32 data32 nopw %cs:0x0(%rax,%rax,1)
```

```
 0.00 :        136f10:       mov    -0x7(%rsi),%edx
 0.00 :        136f13:       mov    -0x4(%rsi),%ecx
 0.00 :        136f16:       mov    %edx,-0x7(%rdi)
 0.00 :        136f19:       mov    %ecx,-0x4(%rdi)
 0.00 :        136f1c:       retq
 0.00 :        136f1d:       nopl   (%rax)
 0.00 :        136f20:       lddqu  -0x86(%rsi),%xmm0
 0.00 :        136f28:       movdqu %xmm0,-0x86(%rdi)
 0.00 :        136f30:       lddqu  -0x76(%rsi),%xmm0
 0.00 :        136f35:       movdqu %xmm0,-0x76(%rdi)
 0.00 :        136f3a:       lddqu  -0x66(%rsi),%xmm0
 0.00 :        136f3f:       movdqu %xmm0,-0x66(%rdi)
 0.00 :        136f44:       lddqu  -0x56(%rsi),%xmm0
 0.00 :        136f49:       movdqu %xmm0,-0x56(%rdi)
 0.00 :        136f4e:       lddqu  -0x46(%rsi),%xmm0
 0.00 :        136f53:       movdqu %xmm0,-0x46(%rdi)
 0.00 :        136f58:       lddqu  -0x36(%rsi),%xmm0
 0.00 :        136f5d:       movdqu %xmm0,-0x36(%rdi)
 0.00 :        136f62:       lddqu  -0x26(%rsi),%xmm0
 0.00 :        136f67:       movdqu %xmm0,-0x26(%rdi)
 0.00 :        136f6c:       lddqu  -0x16(%rsi),%xmm0
 0.00 :        136f71:       lddqu  -0x10(%rsi),%xmm1
 0.00 :        136f76:       movdqu %xmm0,-0x16(%rdi)
 0.00 :        136f7b:       movdqu %xmm1,-0x10(%rdi)
 0.00 :        136f80:       retq
 0.00 :        136f81:       data32 data32 data32 data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :        136f90:       mov    -0x6(%rsi),%edx
 0.00 :        136f93:       mov    -0x4(%rsi),%ecx
 0.00 :        136f96:       mov    %edx,-0x6(%rdi)
 0.00 :        136f99:       mov    %ecx,-0x4(%rdi)
 0.00 :        136f9c:       retq
 0.00 :        136f9d:       nopl   (%rax)
 0.00 :        136fa0:       lddqu  -0x85(%rsi),%xmm0
 0.00 :        136fa8:       movdqu %xmm0,-0x85(%rdi)
 0.00 :        136fb0:       lddqu  -0x75(%rsi),%xmm0
 0.00 :        136fb5:       movdqu %xmm0,-0x75(%rdi)
 0.00 :        136fba:       lddqu  -0x65(%rsi),%xmm0
 0.00 :        136fbf:       movdqu %xmm0,-0x65(%rdi)
 0.00 :        136fc4:       lddqu  -0x55(%rsi),%xmm0
 0.00 :        136fc9:       movdqu %xmm0,-0x55(%rdi)
 0.00 :        136fce:       lddqu  -0x45(%rsi),%xmm0
 0.00 :        136fd3:       movdqu %xmm0,-0x45(%rdi)
 0.00 :        136fd8:       lddqu  -0x35(%rsi),%xmm0
 0.00 :        136fdd:       movdqu %xmm0,-0x35(%rdi)
 0.00 :        136fe2:       lddqu  -0x25(%rsi),%xmm0
 0.00 :        136fe7:       movdqu %xmm0,-0x25(%rdi)
 0.00 :        136fec:       lddqu  -0x15(%rsi),%xmm0
 0.00 :        136ff1:       lddqu  -0x10(%rsi),%xmm1
 0.00 :        136ff6:       movdqu %xmm0,-0x15(%rdi)
 0.00 :        136ffb:       movdqu %xmm1,-0x10(%rdi)
 0.00 :        137000:       retq
 0.00 :        137001:       data32 data32 data32 data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :        137010:       mov    -0x5(%rsi),%edx
 0.00 :        137013:       mov    -0x4(%rsi),%ecx
 0.00 :        137016:       mov    %edx,-0x5(%rdi)
 0.00 :        137019:       mov    %ecx,-0x4(%rdi)
 0.00 :        13701c:       retq
 0.00 :        13701d:       nopl   (%rax)
 0.00 :        137020:       lddqu  -0x84(%rsi),%xmm0
 0.00 :        137028:       movdqu %xmm0,-0x84(%rdi)
 0.00 :        137030:       lddqu  -0x74(%rsi),%xmm0
 0.00 :        137035:       movdqu %xmm0,-0x74(%rdi)
 0.00 :        13703a:       lddqu  -0x64(%rsi),%xmm0
 0.00 :        13703f:       movdqu %xmm0,-0x64(%rdi)
```

```
 0.00 :         137044:       lddqu  -0x54(%rsi),%xmm0
 0.00 :         137049:       movdqu %xmm0,-0x54(%rdi)
 0.00 :         13704e:       lddqu  -0x44(%rsi),%xmm0
 0.00 :         137053:       movdqu %xmm0,-0x44(%rdi)
 0.00 :         137058:       lddqu  -0x34(%rsi),%xmm0
 0.00 :         13705d:       movdqu %xmm0,-0x34(%rdi)
 0.00 :         137062:       lddqu  -0x24(%rsi),%xmm0
 0.00 :         137067:       movdqu %xmm0,-0x24(%rdi)
 0.00 :         13706c:       lddqu  -0x14(%rsi),%xmm0
 0.00 :         137071:       lddqu  -0x10(%rsi),%xmm1
 0.00 :         137076:       movdqu %xmm0,-0x14(%rdi)
 0.00 :         13707b:       movdqu %xmm1,-0x10(%rdi)
 0.00 :         137080:       retq
 0.00 :         137081:       data32 data32 data32 data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :         137090:       mov    -0x4(%rsi),%edx
 0.00 :         137093:       mov    %edx,-0x4(%rdi)
 0.00 :         137096:       retq
 0.00 :         137097:       nopw   0x0(%rax,%rax,1)
 0.00 :         1370a0:       lddqu  -0x83(%rsi),%xmm0
 0.00 :         1370a8:       movdqu %xmm0,-0x83(%rdi)
 0.00 :         1370b0:       lddqu  -0x73(%rsi),%xmm0
 0.00 :         1370b5:       movdqu %xmm0,-0x73(%rdi)
 0.00 :         1370ba:       lddqu  -0x63(%rsi),%xmm0
 0.00 :         1370bf:       movdqu %xmm0,-0x63(%rdi)
 0.00 :         1370c4:       lddqu  -0x53(%rsi),%xmm0
 0.00 :         1370c9:       movdqu %xmm0,-0x53(%rdi)
 0.00 :         1370ce:       lddqu  -0x43(%rsi),%xmm0
 0.00 :         1370d3:       movdqu %xmm0,-0x43(%rdi)
 0.00 :         1370d8:       lddqu  -0x33(%rsi),%xmm0
 0.00 :         1370dd:       movdqu %xmm0,-0x33(%rdi)
 0.00 :         1370e2:       lddqu  -0x23(%rsi),%xmm0
 0.00 :         1370e7:       movdqu %xmm0,-0x23(%rdi)
 0.00 :         1370ec:       lddqu  -0x13(%rsi),%xmm0
 0.00 :         1370f1:       lddqu  -0x10(%rsi),%xmm1
 0.00 :         1370f6:       movdqu %xmm0,-0x13(%rdi)
 0.00 :         1370fb:       movdqu %xmm1,-0x10(%rdi)
 0.00 :         137100:       retq
 0.00 :         137101:       data32 data32 data32 data32 data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :         137110:       mov    -0x3(%rsi),%dx
 0.00 :         137114:       mov    -0x2(%rsi),%cx
 0.00 :         137118:       mov    %dx,-0x3(%rdi)
 0.00 :         13711c:       mov    %cx,-0x2(%rdi)
 0.00 :         137120:       retq
 0.00 :         137121:       data32 data32 data32 data32 data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :         137130:       lddqu  -0x82(%rsi),%xmm0
 0.00 :         137138:       movdqu %xmm0,-0x82(%rdi)
 0.00 :         137140:       lddqu  -0x72(%rsi),%xmm0
 0.00 :         137145:       movdqu %xmm0,-0x72(%rdi)
 0.00 :         13714a:       lddqu  -0x62(%rsi),%xmm0
 0.00 :         13714f:       movdqu %xmm0,-0x62(%rdi)
 0.00 :         137154:       lddqu  -0x52(%rsi),%xmm0
 0.00 :         137159:       movdqu %xmm0,-0x52(%rdi)
 0.00 :         13715e:       lddqu  -0x42(%rsi),%xmm0
 0.00 :         137163:       movdqu %xmm0,-0x42(%rdi)
 0.00 :         137168:       lddqu  -0x32(%rsi),%xmm0
 0.00 :         13716d:       movdqu %xmm0,-0x32(%rdi)
 0.00 :         137172:       lddqu  -0x22(%rsi),%xmm0
 0.00 :         137177:       movdqu %xmm0,-0x22(%rdi)
 0.00 :         13717c:       lddqu  -0x12(%rsi),%xmm0
 0.00 :         137181:       lddqu  -0x10(%rsi),%xmm1
 0.00 :         137186:       movdqu %xmm0,-0x12(%rdi)
 0.00 :         13718b:       movdqu %xmm1,-0x10(%rdi)
 0.00 :         137190:       retq
 0.00 :         137191:       data32 data32 data32 data32 data32 nopw %cs:0x0(%rax,%rax,1)
```

```
 0.00 :         1371a0:       movzwl -0x2(%rsi),%edx
 0.00 :         1371a4:       mov    %dx,-0x2(%rdi)
 0.00 :         1371a8:       retq
 0.00 :         1371a9:       nopl   0x0(%rax)
 0.00 :         1371b0:       lddqu  -0x81(%rsi),%xmm0
 0.00 :         1371b8:       movdqu %xmm0,-0x81(%rdi)
 0.00 :         1371c0:       lddqu  -0x71(%rsi),%xmm0
 0.00 :         1371c5:       movdqu %xmm0,-0x71(%rdi)
 0.00 :         1371ca:       lddqu  -0x61(%rsi),%xmm0
 0.00 :         1371cf:       movdqu %xmm0,-0x61(%rdi)
 0.00 :         1371d4:       lddqu  -0x51(%rsi),%xmm0
 0.00 :         1371d9:       movdqu %xmm0,-0x51(%rdi)
 0.00 :         1371de:       lddqu  -0x41(%rsi),%xmm0
 0.00 :         1371e3:       movdqu %xmm0,-0x41(%rdi)
 0.00 :         1371e8:       lddqu  -0x31(%rsi),%xmm0
 0.00 :         1371ed:       movdqu %xmm0,-0x31(%rdi)
 0.00 :         1371f2:       lddqu  -0x21(%rsi),%xmm0
 0.00 :         1371f7:       movdqu %xmm0,-0x21(%rdi)
 0.00 :         1371fc:       lddqu  -0x11(%rsi),%xmm0
 0.00 :         137201:       lddqu  -0x10(%rsi),%xmm1
 0.00 :         137206:       movdqu %xmm0,-0x11(%rdi)
 0.00 :         13720b:       movdqu %xmm1,-0x10(%rdi)
 0.00 :         137210:       retq
 0.00 :         137211:       data32 data32 data32 data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :         137220:       movzbl -0x1(%rsi),%edx
 0.00 :         137224:       mov    %dl,-0x1(%rdi)
 0.00 :         137227:       retq
 0.00 :         137228:       nopl   0x0(%rax,%rax,1)
 0.00 :         137230:       lddqu  0x70(%rsi),%xmm0
 0.00 :         137235:       movdqu %xmm0,0x70(%rdi)
 0.00 :         13723a:       lddqu  0x60(%rsi),%xmm0
 0.00 :         13723f:       movdqu %xmm0,0x60(%rdi)
 0.00 :         137244:       lddqu  0x50(%rsi),%xmm0
 0.00 :         137249:       movdqu %xmm0,0x50(%rdi)
 0.00 :         13724e:       lddqu  0x40(%rsi),%xmm0
 0.00 :         137253:       movdqu %xmm0,0x40(%rdi)
 0.00 :         137258:       lddqu  0x30(%rsi),%xmm0
 0.00 :         13725d:       movdqu %xmm0,0x30(%rdi)
 0.00 :         137262:       lddqu  0x20(%rsi),%xmm0
 0.00 :         137267:       movdqu %xmm0,0x20(%rdi)
 0.00 :         13726c:       lddqu  0x10(%rsi),%xmm0
 0.00 :         137271:       movdqu %xmm0,0x10(%rdi)
 0.00 :         137276:       lddqu  (%rsi),%xmm0
 0.00 :         13727a:       movdqu %xmm0,(%rdi)
 0.00 :         13727e:       retq
 0.00 :         13727f:       nop
 0.00 :         137280:       lddqu  0x7f(%rsi),%xmm0
 0.00 :         137285:       movdqu %xmm0,0x7f(%rdi)
 0.00 :         13728a:       lddqu  0x6f(%rsi),%xmm0
 1.41 :         13728f:       movdqu %xmm0,0x6f(%rdi)
 0.00 :         137294:       lddqu  0x5f(%rsi),%xmm0
 0.00 :         137299:       movdqu %xmm0,0x5f(%rdi)
 0.00 :         13729e:       lddqu  0x4f(%rsi),%xmm0
 0.00 :         1372a3:       movdqu %xmm0,0x4f(%rdi)
 0.00 :         1372a8:       lddqu  0x3f(%rsi),%xmm0
 0.00 :         1372ad:       movdqu %xmm0,0x3f(%rdi)
 0.00 :         1372b2:       lddqu  0x2f(%rsi),%xmm0
 0.00 :         1372b7:       movdqu %xmm0,0x2f(%rdi)
 0.00 :         1372bc:       lddqu  0x1f(%rsi),%xmm0
 0.00 :         1372c1:       movdqu %xmm0,0x1f(%rdi)
 0.00 :         1372c6:       lddqu  0xf(%rsi),%xmm0
 0.00 :         1372cb:       lddqu  (%rsi),%xmm1
 0.00 :         1372cf:       movdqu %xmm0,0xf(%rdi)
 0.00 :         1372d4:       movdqu %xmm1,(%rdi)
```

```
 0.00 :        1372d8:       retq
 0.00 :        1372d9:       nopl   0x0(%rax)
 0.00 :        1372e0:       mov    0x7(%rsi),%rdx
 0.00 :        1372e4:       mov    (%rsi),%rcx
 0.00 :        1372e7:       mov    %rdx,0x7(%rdi)
 0.00 :        1372eb:       mov    %rcx,(%rdi)
 0.00 :        1372ee:       retq
 0.00 :        1372ef:       nop
 0.00 :        1372f0:       lddqu  0x7e(%rsi),%xmm0
 0.00 :        1372f5:       movdqu %xmm0,0x7e(%rdi)
 0.00 :        1372fa:       lddqu  0x6e(%rsi),%xmm0
 0.00 :        1372ff:       movdqu %xmm0,0x6e(%rdi)
 0.00 :        137304:       lddqu  0x5e(%rsi),%xmm0
 0.00 :        137309:       movdqu %xmm0,0x5e(%rdi)
 0.00 :        13730e:       lddqu  0x4e(%rsi),%xmm0
 0.00 :        137313:       movdqu %xmm0,0x4e(%rdi)
 0.00 :        137318:       lddqu  0x3e(%rsi),%xmm0
 0.00 :        13731d:       movdqu %xmm0,0x3e(%rdi)
 0.00 :        137322:       lddqu  0x2e(%rsi),%xmm0
 0.00 :        137327:       movdqu %xmm0,0x2e(%rdi)
 0.00 :        13732c:       lddqu  0x1e(%rsi),%xmm0
 0.00 :        137331:       movdqu %xmm0,0x1e(%rdi)
 0.00 :        137336:       lddqu  0xe(%rsi),%xmm0
 0.00 :        13733b:       lddqu  (%rsi),%xmm1
 0.00 :        13733f:       movdqu %xmm0,0xe(%rdi)
 0.00 :        137344:       movdqu %xmm1,(%rdi)
 0.00 :        137348:       retq
 0.00 :        137349:       nopl   0x0(%rax)
 0.00 :        137350:       mov    0x6(%rsi),%rdx
 0.00 :        137354:       mov    (%rsi),%rcx
 0.00 :        137357:       mov    %rdx,0x6(%rdi)
 0.00 :        13735b:       mov    %rcx,(%rdi)
 0.00 :        13735e:       retq
 0.00 :        13735f:       nop
 0.00 :        137360:       lddqu  0x7d(%rsi),%xmm0
 0.00 :        137365:       movdqu %xmm0,0x7d(%rdi)
 0.00 :        13736a:       lddqu  0x6d(%rsi),%xmm0
 0.00 :        13736f:       movdqu %xmm0,0x6d(%rdi)
 0.00 :        137374:       lddqu  0x5d(%rsi),%xmm0
 0.00 :        137379:       movdqu %xmm0,0x5d(%rdi)
 0.00 :        13737e:       lddqu  0x4d(%rsi),%xmm0
 0.00 :        137383:       movdqu %xmm0,0x4d(%rdi)
 0.00 :        137388:       lddqu  0x3d(%rsi),%xmm0
 0.00 :        13738d:       movdqu %xmm0,0x3d(%rdi)
 0.00 :        137392:       lddqu  0x2d(%rsi),%xmm0
 0.00 :        137397:       movdqu %xmm0,0x2d(%rdi)
 0.00 :        13739c:       lddqu  0x1d(%rsi),%xmm0
 0.00 :        1373a1:       movdqu %xmm0,0x1d(%rdi)
 0.00 :        1373a6:       lddqu  0xd(%rsi),%xmm0
 0.00 :        1373ab:       lddqu  (%rsi),%xmm1
 0.00 :        1373af:       movdqu %xmm0,0xd(%rdi)
 0.00 :        1373b4:       movdqu %xmm1,(%rdi)
 0.00 :        1373b8:       retq
 0.00 :        1373b9:       nopl   0x0(%rax)
 0.00 :        1373c0:       mov    0x5(%rsi),%rdx
 0.00 :        1373c4:       mov    (%rsi),%rcx
 0.00 :        1373c7:       mov    %rdx,0x5(%rdi)
 0.00 :        1373cb:       mov    %rcx,(%rdi)
 0.00 :        1373ce:       retq
 0.00 :        1373cf:       nop
 0.00 :        1373d0:       lddqu  0x7c(%rsi),%xmm0
 0.00 :        1373d5:       movdqu %xmm0,0x7c(%rdi)
 0.00 :        1373da:       lddqu  0x6c(%rsi),%xmm0
 0.00 :        1373df:       movdqu %xmm0,0x6c(%rdi)
```

```
 0.00 :        1373e4:       lddqu  0x5c(%rsi),%xmm0
 0.00 :        1373e9:       movdqu %xmm0,0x5c(%rdi)
 0.00 :        1373ee:       lddqu  0x4c(%rsi),%xmm0
 0.00 :        1373f3:       movdqu %xmm0,0x4c(%rdi)
 0.00 :        1373f8:       lddqu  0x3c(%rsi),%xmm0
 0.00 :        1373fd:       movdqu %xmm0,0x3c(%rdi)
 0.00 :        137402:       lddqu  0x2c(%rsi),%xmm0
 0.00 :        137407:       movdqu %xmm0,0x2c(%rdi)
 0.00 :        13740c:       lddqu  0x1c(%rsi),%xmm0
 0.00 :        137411:       movdqu %xmm0,0x1c(%rdi)
 0.00 :        137416:       lddqu  0xc(%rsi),%xmm0
 0.00 :        13741b:       lddqu  (%rsi),%xmm1
 0.00 :        13741f:       movdqu %xmm0,0xc(%rdi)
 0.00 :        137424:       movdqu %xmm1,(%rdi)
 0.00 :        137428:       retq
 0.00 :        137429:       nopl   0x0(%rax)
 0.00 :        137430:       mov    0x4(%rsi),%rdx
 0.00 :        137434:       mov    (%rsi),%rcx
 0.00 :        137437:       mov    %rdx,0x4(%rdi)
 0.00 :        13743b:       mov    %rcx,(%rdi)
 0.00 :        13743e:       retq
 0.00 :        13743f:       nop
 0.00 :        137440:       lddqu  0x7b(%rsi),%xmm0
 0.00 :        137445:       movdqu %xmm0,0x7b(%rdi)
 0.00 :        13744a:       lddqu  0x6b(%rsi),%xmm0
 0.00 :        13744f:       movdqu %xmm0,0x6b(%rdi)
 0.00 :        137454:       lddqu  0x5b(%rsi),%xmm0
 0.00 :        137459:       movdqu %xmm0,0x5b(%rdi)
 0.00 :        13745e:       lddqu  0x4b(%rsi),%xmm0
 0.00 :        137463:       movdqu %xmm0,0x4b(%rdi)
 0.00 :        137468:       lddqu  0x3b(%rsi),%xmm0
 0.00 :        13746d:       movdqu %xmm0,0x3b(%rdi)
 0.00 :        137472:       lddqu  0x2b(%rsi),%xmm0
 0.00 :        137477:       movdqu %xmm0,0x2b(%rdi)
 0.00 :        13747c:       lddqu  0x1b(%rsi),%xmm0
 0.00 :        137481:       movdqu %xmm0,0x1b(%rdi)
 0.00 :        137486:       lddqu  0xb(%rsi),%xmm0
 0.00 :        13748b:       lddqu  (%rsi),%xmm1
 0.00 :        13748f:       movdqu %xmm0,0xb(%rdi)
 0.00 :        137494:       movdqu %xmm1,(%rdi)
 0.00 :        137498:       retq
 0.00 :        137499:       nopl   0x0(%rax)
 0.00 :        1374a0:       mov    0x3(%rsi),%rdx
 0.00 :        1374a4:       mov    (%rsi),%rcx
 0.00 :        1374a7:       mov    %rdx,0x3(%rdi)
 0.00 :        1374ab:       mov    %rcx,(%rdi)
 0.00 :        1374ae:       retq
 0.00 :        1374af:       nop
 0.00 :        1374b0:       lddqu  0x7a(%rsi),%xmm0
 0.00 :        1374b5:       movdqu %xmm0,0x7a(%rdi)
 0.00 :        1374ba:       lddqu  0x6a(%rsi),%xmm0
 0.00 :        1374bf:       movdqu %xmm0,0x6a(%rdi)
 0.00 :        1374c4:       lddqu  0x5a(%rsi),%xmm0
 0.00 :        1374c9:       movdqu %xmm0,0x5a(%rdi)
 0.00 :        1374ce:       lddqu  0x4a(%rsi),%xmm0
 0.00 :        1374d3:       movdqu %xmm0,0x4a(%rdi)
 0.00 :        1374d8:       lddqu  0x3a(%rsi),%xmm0
 0.00 :        1374dd:       movdqu %xmm0,0x3a(%rdi)
 0.00 :        1374e2:       lddqu  0x2a(%rsi),%xmm0
 0.00 :        1374e7:       movdqu %xmm0,0x2a(%rdi)
 0.00 :        1374ec:       lddqu  0x1a(%rsi),%xmm0
 0.00 :        1374f1:       movdqu %xmm0,0x1a(%rdi)
 0.00 :        1374f6:       lddqu  0xa(%rsi),%xmm0
 0.00 :        1374fb:       lddqu  (%rsi),%xmm1
```

```
 0.00 :        1374ff:       movdqu %xmm0,0xa(%rdi)
 0.00 :        137504:       movdqu %xmm1,(%rdi)
 0.00 :        137508:       retq
 0.00 :        137509:       nopl   0x0(%rax)
 0.00 :        137510:       mov    0x2(%rsi),%rdx
 0.00 :        137514:       mov    (%rsi),%rcx
 0.00 :        137517:       mov    %rdx,0x2(%rdi)
 0.00 :        13751b:       mov    %rcx,(%rdi)
 0.00 :        13751e:       retq
 0.00 :        13751f:       nop
 0.00 :        137520:       lddqu  0x79(%rsi),%xmm0
 0.00 :        137525:       movdqu %xmm0,0x79(%rdi)
 0.00 :        13752a:       lddqu  0x69(%rsi),%xmm0
 0.00 :        13752f:       movdqu %xmm0,0x69(%rdi)
 0.00 :        137534:       lddqu  0x59(%rsi),%xmm0
 0.00 :        137539:       movdqu %xmm0,0x59(%rdi)
 0.00 :        13753e:       lddqu  0x49(%rsi),%xmm0
 0.00 :        137543:       movdqu %xmm0,0x49(%rdi)
 0.00 :        137548:       lddqu  0x39(%rsi),%xmm0
 0.00 :        13754d:       movdqu %xmm0,0x39(%rdi)
 0.00 :        137552:       lddqu  0x29(%rsi),%xmm0
 0.00 :        137557:       movdqu %xmm0,0x29(%rdi)
 0.00 :        13755c:       lddqu  0x19(%rsi),%xmm0
 0.00 :        137561:       movdqu %xmm0,0x19(%rdi)
 0.00 :        137566:       lddqu  0x9(%rsi),%xmm0
 0.00 :        13756b:       lddqu  (%rsi),%xmm1
 0.00 :        13756f:       movdqu %xmm0,0x9(%rdi)
 0.00 :        137574:       movdqu %xmm1,(%rdi)
 0.00 :        137578:       retq
 0.00 :        137579:       nopl   0x0(%rax)
 0.00 :        137580:       mov    0x1(%rsi),%rdx
 0.00 :        137584:       mov    (%rsi),%rcx
 0.00 :        137587:       mov    %rdx,0x1(%rdi)
 0.00 :        13758b:       mov    %rcx,(%rdi)
 0.00 :        13758e:       retq
 0.00 :        13758f:       nop
 0.00 :        137590:       lddqu  0x78(%rsi),%xmm0
 0.00 :        137595:       movdqu %xmm0,0x78(%rdi)
 0.00 :        13759a:       lddqu  0x68(%rsi),%xmm0
 0.00 :        13759f:       movdqu %xmm0,0x68(%rdi)
 0.00 :        1375a4:       lddqu  0x58(%rsi),%xmm0
 0.00 :        1375a9:       movdqu %xmm0,0x58(%rdi)
 0.00 :        1375ae:       lddqu  0x48(%rsi),%xmm0
 0.00 :        1375b3:       movdqu %xmm0,0x48(%rdi)
 0.00 :        1375b8:       lddqu  0x38(%rsi),%xmm0
 0.00 :        1375bd:       movdqu %xmm0,0x38(%rdi)
 0.00 :        1375c2:       lddqu  0x28(%rsi),%xmm0
 0.00 :        1375c7:       movdqu %xmm0,0x28(%rdi)
 0.00 :        1375cc:       lddqu  0x18(%rsi),%xmm0
 0.00 :        1375d1:       movdqu %xmm0,0x18(%rdi)
 0.00 :        1375d6:       lddqu  0x8(%rsi),%xmm0
 0.00 :        1375db:       lddqu  (%rsi),%xmm1
 0.00 :        1375df:       movdqu %xmm0,0x8(%rdi)
 0.00 :        1375e4:       movdqu %xmm1,(%rdi)
 0.00 :        1375e8:       retq
 0.00 :        1375e9:       nopl   0x0(%rax)
 0.00 :        1375f0:       mov    (%rsi),%rdx
 0.00 :        1375f3:       mov    %rdx,(%rdi)
 0.00 :        1375f6:       retq
 0.00 :        1375f7:       nopw   0x0(%rax,%rax,1)
 0.00 :        137600:       lddqu  0x77(%rsi),%xmm0
 0.00 :        137605:       movdqu %xmm0,0x77(%rdi)
 0.00 :        13760a:       lddqu  0x67(%rsi),%xmm0
 0.00 :        13760f:       movdqu %xmm0,0x67(%rdi)
```

```
 0.00 :        137614:       lddqu  0x57(%rsi),%xmm0
 0.00 :        137619:       movdqu %xmm0,0x57(%rdi)
 0.00 :        13761e:       lddqu  0x47(%rsi),%xmm0
 0.00 :        137623:       movdqu %xmm0,0x47(%rdi)
 0.00 :        137628:       lddqu  0x37(%rsi),%xmm0
 0.00 :        13762d:       movdqu %xmm0,0x37(%rdi)
 0.00 :        137632:       lddqu  0x27(%rsi),%xmm0
 0.00 :        137637:       movdqu %xmm0,0x27(%rdi)
 0.00 :        13763c:       lddqu  0x17(%rsi),%xmm0
 0.00 :        137641:       movdqu %xmm0,0x17(%rdi)
 0.00 :        137646:       lddqu  0x7(%rsi),%xmm0
 0.00 :        13764b:       lddqu  (%rsi),%xmm1
 0.00 :        13764f:       movdqu %xmm0,0x7(%rdi)
 0.00 :        137654:       movdqu %xmm1,(%rdi)
 0.00 :        137658:       retq
 0.00 :        137659:       nopl   0x0(%rax)
 0.00 :        137660:       mov    0x3(%rsi),%edx
 0.00 :        137663:       mov    (%rsi),%ecx
 0.00 :        137665:       mov    %edx,0x3(%rdi)
 0.00 :        137668:       mov    %ecx,(%rdi)
 0.00 :        13766a:       retq
 0.00 :        13766b:       nopl   0x0(%rax,%rax,1)
 0.00 :        137670:       lddqu  0x76(%rsi),%xmm0
 0.00 :        137675:       movdqu %xmm0,0x76(%rdi)
 0.00 :        13767a:       lddqu  0x66(%rsi),%xmm0
 0.00 :        13767f:       movdqu %xmm0,0x66(%rdi)
 0.00 :        137684:       lddqu  0x56(%rsi),%xmm0
 0.00 :        137689:       movdqu %xmm0,0x56(%rdi)
 0.00 :        13768e:       lddqu  0x46(%rsi),%xmm0
 0.00 :        137693:       movdqu %xmm0,0x46(%rdi)
 0.00 :        137698:       lddqu  0x36(%rsi),%xmm0
 0.00 :        13769d:       movdqu %xmm0,0x36(%rdi)
 0.00 :        1376a2:       lddqu  0x26(%rsi),%xmm0
 0.00 :        1376a7:       movdqu %xmm0,0x26(%rdi)
 0.00 :        1376ac:       lddqu  0x16(%rsi),%xmm0
 0.00 :        1376b1:       movdqu %xmm0,0x16(%rdi)
 0.00 :        1376b6:       lddqu  0x6(%rsi),%xmm0
 0.00 :        1376bb:       lddqu  (%rsi),%xmm1
 0.00 :        1376bf:       movdqu %xmm0,0x6(%rdi)
 0.00 :        1376c4:       movdqu %xmm1,(%rdi)
 0.00 :        1376c8:       retq
 0.00 :        1376c9:       nopl   0x0(%rax)
 0.00 :        1376d0:       mov    0x2(%rsi),%edx
 0.00 :        1376d3:       mov    (%rsi),%ecx
 0.00 :        1376d5:       mov    %edx,0x2(%rdi)
 0.00 :        1376d8:       mov    %ecx,(%rdi)
 0.00 :        1376da:       retq
 0.00 :        1376db:       nopl   0x0(%rax,%rax,1)
 0.00 :        1376e0:       lddqu  0x75(%rsi),%xmm0
 0.00 :        1376e5:       movdqu %xmm0,0x75(%rdi)
 0.00 :        1376ea:       lddqu  0x65(%rsi),%xmm0
 0.00 :        1376ef:       movdqu %xmm0,0x65(%rdi)
 0.00 :        1376f4:       lddqu  0x55(%rsi),%xmm0
 0.00 :        1376f9:       movdqu %xmm0,0x55(%rdi)
 0.00 :        1376fe:       lddqu  0x45(%rsi),%xmm0
 0.00 :        137703:       movdqu %xmm0,0x45(%rdi)
 0.00 :        137708:       lddqu  0x35(%rsi),%xmm0
 0.00 :        13770d:       movdqu %xmm0,0x35(%rdi)
 0.00 :        137712:       lddqu  0x25(%rsi),%xmm0
 0.00 :        137717:       movdqu %xmm0,0x25(%rdi)
 0.00 :        13771c:       lddqu  0x15(%rsi),%xmm0
 0.00 :        137721:       movdqu %xmm0,0x15(%rdi)
 0.00 :        137726:       lddqu  0x5(%rsi),%xmm0
 0.00 :        13772b:       lddqu  (%rsi),%xmm1
```

```
 0.00 :        13772f:       movdqu %xmm0,0x5(%rdi)
 0.00 :        137734:       movdqu %xmm1,(%rdi)
 0.00 :        137738:       retq
 0.00 :        137739:       nopl   0x0(%rax)
 0.00 :        137740:       mov    0x1(%rsi),%edx
 0.00 :        137743:       mov    (%rsi),%ecx
 0.00 :        137745:       mov    %edx,0x1(%rdi)
 0.00 :        137748:       mov    %ecx,(%rdi)
 0.00 :        13774a:       retq
 0.00 :        13774b:       nopl   0x0(%rax,%rax,1)
 0.00 :        137750:       lddqu  0x74(%rsi),%xmm0
 0.00 :        137755:       movdqu %xmm0,0x74(%rdi)
 0.00 :        13775a:       lddqu  0x64(%rsi),%xmm0
 0.00 :        13775f:       movdqu %xmm0,0x64(%rdi)
 0.00 :        137764:       lddqu  0x54(%rsi),%xmm0
 0.00 :        137769:       movdqu %xmm0,0x54(%rdi)
 0.00 :        13776e:       lddqu  0x44(%rsi),%xmm0
 0.00 :        137773:       movdqu %xmm0,0x44(%rdi)
 0.00 :        137778:       lddqu  0x34(%rsi),%xmm0
 0.00 :        13777d:       movdqu %xmm0,0x34(%rdi)
 0.00 :        137782:       lddqu  0x24(%rsi),%xmm0
 0.00 :        137787:       movdqu %xmm0,0x24(%rdi)
 0.00 :        13778c:       lddqu  0x14(%rsi),%xmm0
 0.00 :        137791:       movdqu %xmm0,0x14(%rdi)
 0.00 :        137796:       lddqu  0x4(%rsi),%xmm0
 0.00 :        13779b:       lddqu  (%rsi),%xmm1
 0.00 :        13779f:       movdqu %xmm0,0x4(%rdi)
 0.00 :        1377a4:       movdqu %xmm1,(%rdi)
 0.00 :        1377a8:       retq
 0.00 :        1377a9:       nopl   0x0(%rax)
 0.00 :        1377b0:       mov    (%rsi),%edx
 0.00 :        1377b2:       mov    %edx,(%rdi)
 0.00 :        1377b4:       retq
 0.00 :        1377b5:       data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :        1377c0:       lddqu  0x73(%rsi),%xmm0
 0.00 :        1377c5:       movdqu %xmm0,0x73(%rdi)
 0.00 :        1377ca:       lddqu  0x63(%rsi),%xmm0
 0.00 :        1377cf:       movdqu %xmm0,0x63(%rdi)
 0.00 :        1377d4:       lddqu  0x53(%rsi),%xmm0
 0.00 :        1377d9:       movdqu %xmm0,0x53(%rdi)
 0.00 :        1377de:       lddqu  0x43(%rsi),%xmm0
 0.00 :        1377e3:       movdqu %xmm0,0x43(%rdi)
 0.00 :        1377e8:       lddqu  0x33(%rsi),%xmm0
 0.00 :        1377ed:       movdqu %xmm0,0x33(%rdi)
 0.00 :        1377f2:       lddqu  0x23(%rsi),%xmm0
 0.00 :        1377f7:       movdqu %xmm0,0x23(%rdi)
 0.00 :        1377fc:       lddqu  0x13(%rsi),%xmm0
 0.00 :        137801:       movdqu %xmm0,0x13(%rdi)
 0.00 :        137806:       lddqu  0x3(%rsi),%xmm0
 0.00 :        13780b:       lddqu  (%rsi),%xmm1
 0.00 :        13780f:       movdqu %xmm0,0x3(%rdi)
 0.00 :        137814:       movdqu %xmm1,(%rdi)
 0.00 :        137818:       retq
 0.00 :        137819:       nopl   0x0(%rax)
 0.00 :        137820:       mov    0x1(%rsi),%dx
 0.00 :        137824:       mov    (%rsi),%cx
 0.00 :        137827:       mov    %dx,0x1(%rdi)
 0.00 :        13782b:       mov    %cx,(%rdi)
 0.00 :        13782e:       retq
 0.00 :        13782f:       nop
 0.00 :        137830:       lddqu  0x72(%rsi),%xmm0
 0.00 :        137835:       movdqu %xmm0,0x72(%rdi)
 0.00 :        13783a:       lddqu  0x62(%rsi),%xmm0
 0.00 :        13783f:       movdqu %xmm0,0x62(%rdi)
```

```
    0.00 :          137844:       lddqu  0x52(%rsi),%xmm0
    0.00 :          137849:       movdqu %xmm0,0x52(%rdi)
    0.00 :          13784e:       lddqu  0x42(%rsi),%xmm0
    0.00 :          137853:       movdqu %xmm0,0x42(%rdi)
    0.00 :          137858:       lddqu  0x32(%rsi),%xmm0
    0.00 :          13785d:       movdqu %xmm0,0x32(%rdi)
    0.00 :          137862:       lddqu  0x22(%rsi),%xmm0
    0.00 :          137867:       movdqu %xmm0,0x22(%rdi)
    0.00 :          13786c:       lddqu  0x12(%rsi),%xmm0
    0.00 :          137871:       movdqu %xmm0,0x12(%rdi)
    0.00 :          137876:       lddqu  0x2(%rsi),%xmm0
    0.00 :          13787b:       lddqu  (%rsi),%xmm1
    0.00 :          13787f:       movdqu %xmm0,0x2(%rdi)
    0.00 :          137884:       movdqu %xmm1,(%rdi)
    0.00 :          137888:       retq
    0.00 :          137889:       nopl   0x0(%rax)
    0.00 :          137890:       movzwl (%rsi),%edx
    0.00 :          137893:       mov    %dx,(%rdi)
    0.00 :          137896:       retq
    0.00 :          137897:       nopw   0x0(%rax,%rax,1)
    0.00 :          1378a0:       lddqu  0x71(%rsi),%xmm0
    0.00 :          1378a5:       movdqu %xmm0,0x71(%rdi)
    0.00 :          1378aa:       lddqu  0x61(%rsi),%xmm0
    0.00 :          1378af:       movdqu %xmm0,0x61(%rdi)
    0.00 :          1378b4:       lddqu  0x51(%rsi),%xmm0
    0.00 :          1378b9:       movdqu %xmm0,0x51(%rdi)
    0.00 :          1378be:       lddqu  0x41(%rsi),%xmm0
    0.00 :          1378c3:       movdqu %xmm0,0x41(%rdi)
    0.00 :          1378c8:       lddqu  0x31(%rsi),%xmm0
    0.00 :          1378cd:       movdqu %xmm0,0x31(%rdi)
    0.00 :          1378d2:       lddqu  0x21(%rsi),%xmm0
    0.00 :          1378d7:       movdqu %xmm0,0x21(%rdi)
    0.00 :          1378dc:       lddqu  0x11(%rsi),%xmm0
    0.00 :          1378e1:       movdqu %xmm0,0x11(%rdi)
    0.00 :          1378e6:       lddqu  0x1(%rsi),%xmm0
    0.00 :          1378eb:       lddqu  (%rsi),%xmm1
    0.00 :          1378ef:       movdqu %xmm0,0x1(%rdi)
    0.00 :          1378f4:       movdqu %xmm1,(%rdi)
    0.00 :          1378f8:       retq
    0.00 :          1378f9:       nopl   0x0(%rax)
    0.00 :          137900:       movzbl (%rsi),%edx
    0.00 :          137903:       mov    %dl,(%rdi)
    0.00 :          137905:       retq
 Percent |       Source code & Disassembly of libQtGui.so.4.8.4
-----------------------------------------------
         :
         :
         :
         :       Disassembly of section .text:
         :
         :       000000000027ca20 <QImage::scanLine(int) const>:
   47.62 :          27ca20:       mov    0x10(%rdi),%rdx
    0.00 :          27ca24:       test   %rdx,%rdx
    0.00 :          27ca27:       je     27ca40 <QImage::scanLine(int) const+0x20>
    0.00 :          27ca29:       mov    0x34(%rdx),%eax
   30.16 :          27ca2c:       imul   %esi,%eax
   12.70 :          27ca2f:       cltq
    0.00 :          27ca31:       add    0x20(%rdx),%rax
    9.52 :          27ca35:       retq
    0.00 :          27ca36:       nopw   %cs:0x0(%rax,%rax,1)
    0.00 :          27ca40:       xor    %eax,%eax
    0.00 :          27ca42:       retq
 Percent |       Source code & Disassembly of libm-2.17.so
-----------------------------------------------
```

```
         :
         :
         :
         :        Disassembly of section .plt:
         :
         :        0000000000005610 <*ABS*+0x15ae0@plt>:
  100.00 :          5610:       jmpq   *0x2f7a8a(%rip)        # 2fd0a0 <__signbitl+0x2be370>
    0.00 :          5616:       pushq  $0x11
    0.00 :          561b:       jmpq   54f0 <GLIBC_2.15+0x54f0>
 Percent |        Source code & Disassembly of libQtGui.so.4.8.4
-----------------------------------------------
         :
         :
         :
         :        Disassembly of section .text:
         :
         :        000000000027c7c0 <QImage::width() const>:
    2.70 :          27c7c0:       mov    0x10(%rdi),%rax
   78.38 :          27c7c4:       test   %rax,%rax
    0.00 :          27c7c7:       je     27c7d0 <QImage::width() const+0x10>
    2.70 :          27c7c9:       mov    0x4(%rax),%eax
   16.22 :          27c7cc:       retq
    0.00 :          27c7cd:       nopl   (%rax)
    0.00 :          27c7d0:       xor    %eax,%eax
    0.00 :          27c7d2:       retq
 Percent |        Source code & Disassembly of test_harness
-----------------------------------------------
         :
         :
         :
         :        Disassembly of section .plt:
         :
         :        0000000000400f90 <QImage::pixel(int, int) const@plt>:
  100.00 :          400f90:       jmpq   *0x20215a(%rip)        # 6030f0 <_GLOBAL_OFFSET_TABLE_+0xf0>
    0.00 :          400f96:       pushq  $0x1b
    0.00 :          400f9b:       jmpq   400dd0 <_init+0x20>
 Percent |        Source code & Disassembly of test_harness
-----------------------------------------------
         :
         :
         :
         :        Disassembly of section .plt:
         :
         :        0000000000400e00 <ceil@plt>:
  100.00 :          400e00:       jmpq   *0x202222(%rip)        # 603028 <_GLOBAL_OFFSET_TABLE_+0x28>
    0.00 :          400e06:       pushq  $0x2
    0.00 :          400e0b:       jmpq   400dd0 <_init+0x20>
 Percent |        Source code & Disassembly of libQtGui.so.4.8.4
-----------------------------------------------
         :
         :
         :
         :        Disassembly of section .text:
         :
         :        00000000001c1a30 <qt_convert_rgb888_to_rgb32_ssse3(unsigned int*, unsigned char const*, int)>:
    0.00 :          1c1a30:       movslq %edx,%rax
    0.00 :          1c1a33:       mov    %edx,%ecx
    0.00 :          1c1a35:       push   %rbx
    0.00 :          1c1a36:       lea    (%rdi,%rax,4),%r10
    0.00 :          1c1a3a:       mov    %rdi,%rax
    0.00 :          1c1a3d:       shr    $0x2,%rax
    0.00 :          1c1a41:       neg    %eax
    0.00 :          1c1a43:       and    $0x3,%eax
    0.00 :          1c1a46:       cmp    %edx,%eax
```

```
    0.00 :          1c1a48:       cmovle %eax,%ecx
    0.00 :          1c1a4b:       test   %ecx,%ecx
    0.00 :          1c1a4d:       jle    1c1a93 <qt_convert_rgb888_to_rgb32_ssse3(unsigned int*, unsigned char const*,
int)+0x63>
    0.00 :          1c1a4f:       lea    -0x1(%rcx),%ebx
    0.00 :          1c1a52:       mov    %rdi,%r9
    0.00 :          1c1a55:       add    $0x1,%rbx
    0.00 :          1c1a59:       lea    (%rbx,%rbx,2),%r11
    0.00 :          1c1a5d:       add    %rsi,%r11
    0.00 :          1c1a60:       movzbl (%rsi),%eax
    0.00 :          1c1a63:       movzbl 0x2(%rsi),%r8d
    0.00 :          1c1a68:       shl    $0x10,%eax
    0.00 :          1c1a6b:       or     %r8d,%eax
    0.00 :          1c1a6e:       movzbl 0x1(%rsi),%r8d
    0.00 :          1c1a73:       add    $0x3,%rsi
    0.00 :          1c1a77:       shl    $0x8,%r8d
    0.00 :          1c1a7b:       or     %r8d,%eax
    0.00 :          1c1a7e:       or     $0xff000000,%eax
    0.00 :          1c1a83:       mov    %eax,(%r9)
    0.00 :          1c1a86:       add    $0x4,%r9
    0.00 :          1c1a8a:       cmp    %r11,%rsi
    0.00 :          1c1a8d:       jne    1c1a60 <qt_convert_rgb888_to_rgb32_ssse3(unsigned int*, unsigned char const*,
int)+0x30>
    0.00 :          1c1a8f:       lea    (%rdi,%rbx,4),%rdi
    0.00 :          1c1a93:       sub    %ecx,%edx
    0.00 :          1c1a95:       lea    0xf(%rdx),%eax
    0.00 :          1c1a98:       test   %edx,%edx
    0.00 :          1c1a9a:       cmovs  %eax,%edx
    0.00 :          1c1a9d:       sar    $0x4,%edx
    0.00 :          1c1aa0:       test   %edx,%edx
    0.00 :          1c1aa2:       jle    1c1b83 <qt_convert_rgb888_to_rgb32_ssse3(unsigned int*, unsigned char const*,
int)+0x153>
    0.00 :          1c1aa8:       sub    $0x1,%edx
    0.00 :          1c1aab:       movdqa 0x67833d(%rip),%xmm1        # 839df0 <typeinfo name for QDashStroker+0x1d0>
    0.00 :          1c1ab3:       lea    0x1(%rdx),%rcx
    0.00 :          1c1ab7:       mov    %rdi,%rax
    0.00 :          1c1aba:       movdqa 0x6781ee(%rip),%xmm0        # 839cb0 <typeinfo name for QDashStroker+0x90>
    0.00 :          1c1ac2:       lea    (%rcx,%rcx,2),%rdx
    0.00 :          1c1ac6:       movdqa 0x678332(%rip),%xmm4        # 839e00 <typeinfo name for QDashStroker+0x1e0>
    0.00 :          1c1ace:       shl    $0x4,%rdx
    0.00 :          1c1ad2:       add    %rsi,%rdx
    0.00 :          1c1ad5:       nopl   (%rax)
    0.00 :          1c1ad8:       lddqu  (%rsi),%xmm2
   12.90 :          1c1adc:       movdqa %xmm2,%xmm3
    0.00 :          1c1ae0:       pshufb %xmm1,%xmm3
    0.00 :          1c1ae5:       por    %xmm0,%xmm3
    0.00 :          1c1ae9:       movdqa %xmm3,(%rax)
    9.68 :          1c1aed:       lddqu  0x10(%rsi),%xmm3
    9.68 :          1c1af2:       movdqa %xmm3,%xmm5
    0.00 :          1c1af6:       palignr $0xc,%xmm2,%xmm5
    0.00 :          1c1afc:       movdqa %xmm5,%xmm2
    3.23 :          1c1b00:       pshufb %xmm1,%xmm2
    0.00 :          1c1b05:       por    %xmm0,%xmm2
    0.00 :          1c1b09:       movdqa %xmm2,0x10(%rax)
    0.00 :          1c1b0e:       lddqu  0x20(%rsi),%xmm2
    6.45 :          1c1b13:       add    $0x30,%rsi
    0.00 :          1c1b17:       movdqa %xmm2,%xmm5
    3.23 :          1c1b1b:       pshufb %xmm4,%xmm2
    0.00 :          1c1b20:       palignr $0x8,%xmm3,%xmm5
    3.23 :          1c1b26:       movdqa %xmm5,%xmm3
    0.00 :          1c1b2a:       pshufb %xmm1,%xmm3
    0.00 :          1c1b2f:       por    %xmm0,%xmm2
    3.23 :          1c1b33:       por    %xmm0,%xmm3
   32.26 :          1c1b37:       movdqa %xmm2,0x30(%rax)
```

```
   16.13 :          1c1b3c:       movdqa %xmm3,0x20(%rax)
    0.00 :          1c1b41:       add    $0x40,%rax
    0.00 :          1c1b45:       cmp    %rsi,%rdx
    0.00 :          1c1b48:       jne    1c1ad8 <qt_convert_rgb888_to_rgb32_ssse3(unsigned int*, unsigned char const*,
int)+0xa8>
    0.00 :          1c1b4a:       shl    $0x6,%rcx
    0.00 :          1c1b4e:       add    %rdi,%rcx
    0.00 :          1c1b51:       jmp    1c1b7c <qt_convert_rgb888_to_rgb32_ssse3(unsigned int*, unsigned char const*,
int)+0x14c>
    0.00 :          1c1b53:       nopl   0x0(%rax,%rax,1)
    0.00 :          1c1b58:       movzbl (%rdx),%eax
    0.00 :          1c1b5b:       movzbl 0x2(%rdx),%esi
    0.00 :          1c1b5f:       shl    $0x10,%eax
    0.00 :          1c1b62:       or     %esi,%eax
    0.00 :          1c1b64:       movzbl 0x1(%rdx),%esi
    0.00 :          1c1b68:       add    $0x3,%rdx
    0.00 :          1c1b6c:       shl    $0x8,%esi
    0.00 :          1c1b6f:       or     %esi,%eax
    0.00 :          1c1b71:       or     $0xff000000,%eax
    0.00 :          1c1b76:       mov    %eax,(%rcx)
    0.00 :          1c1b78:       add    $0x4,%rcx
    0.00 :          1c1b7c:       cmp    %rcx,%r10
    0.00 :          1c1b7f:       jne    1c1b58 <qt_convert_rgb888_to_rgb32_ssse3(unsigned int*, unsigned char const*,
int)+0x128>
    0.00 :          1c1b81:       pop    %rbx
    0.00 :          1c1b82:       retq
    0.00 :          1c1b83:       mov    %rdi,%rcx
    0.00 :          1c1b86:       mov    %rsi,%rdx
    0.00 :          1c1b89:       jmp    1c1b7c <qt_convert_rgb888_to_rgb32_ssse3(unsigned int*, unsigned char const*,
int)+0x14c>
 Percent |       Source code & Disassembly of test_harness
------------------------------------------------
         :
         :
         :
         :       Disassembly of section .text:
         :
         :       0000000000402410 <Model::prepStraightLine()>:
         :
         :           listLines = new vector<pair <QPoint, QPoint> >* [2];
         :           listAux = new vector<pair <QPoint, QPoint> >* [2];
         :       }
         :
         :       void Model::prepStraightLine() {
    0.00 :         402410:       push   %r15
    0.00 :         402412:       mov    %rdi,%r15
    0.00 :         402415:       push   %r14
    0.00 :         402417:       push   %r13
    0.00 :         402419:       push   %r12
         :           QRgb red = qRgb(255, 0, 0);
         :           QRgb blue = qRgb(0, 0, 255);
         :
         :           for(unsigned int k=0; k<listLines[0]->size(); ++k) {
    0.00 :         40241b:       xor    %r12d,%r12d
         :
         :           listLines = new vector<pair <QPoint, QPoint> >* [2];
         :           listAux = new vector<pair <QPoint, QPoint> >* [2];
         :       }
         :
         :       void Model::prepStraightLine() {
    0.00 :         40241e:       push   %rbp
         :           QRgb red = qRgb(255, 0, 0);
         :           QRgb blue = qRgb(0, 0, 255);
         :
```

```
      :             for(unsigned int k=0; k<listLines[0]->size(); ++k) {
 0.00 :   40241f:       xor    %ebp,%ebp
      :
      :             listLines = new vector<pair <QPoint, QPoint> >* [2];
      :             listAux = new vector<pair <QPoint, QPoint> >* [2];
      :         }
      :
      :         void Model::prepStraightLine() {
 0.00 :   402421:       push   %rbx
 0.00 :   402422:       sub    $0x78,%rsp
      :             QRgb red = qRgb(255, 0, 0);
      :             QRgb blue = qRgb(0, 0, 255);
      :
      :             for(unsigned int k=0; k<listLines[0]->size(); ++k) {
 0.00 :   402426:       mov    0x8(%rdi),%rdx
 0.00 :   40242a:       movsd  0x51e(%rip),%xmm0        # 402950 <_IO_stdin_used+0xa0>
 0.00 :   402432:       mov    (%rdx),%rcx
 0.00 :   402435:       movsd  %xmm0,0x58(%rsp)
      :
      :                 // [23.2.4.2] capacity
      :                 /**  Returns the number of elements in the %vector.  */
      :                 size_type
      :                 size() const _GLIBCXX_NOEXCEPT
      :                 { return size_type(this->_M_impl._M_finish - this->_M_impl._M_start); }
 0.00 :   40243b:       mov    0x8(%rcx),%rax
 0.00 :   40243f:       sub    (%rcx),%rax
 0.00 :   402442:       sar    $0x4,%rax
 0.00 :   402446:       test   %rax,%rax
 0.00 :   402449:       je     4026c6 <Model::prepStraightLine()+0x2b6>
 0.00 :   40244f:       nop
      :                  *  out_of_range lookups are not defined. (For checked lookups
      :                  *  see at().)
      :                  */
      :                 reference
      :                 operator[](size_type __n)
      :                 { return *(this->_M_impl._M_start + __n); }
 0.00 :   402450:       mov    %rbp,%r13
 0.00 :   402453:       xor    %ebx,%ebx
 0.00 :   402455:       movq   $0x0,0x50(%rsp)
 0.00 :   40245e:       shl    $0x4,%r13
 0.00 :   402462:       movq   $0x0,0x48(%rsp)
 0.00 :   40246b:       movq   $0x0,0x40(%rsp)
 0.00 :   402474:       movq   $0x0,0x38(%rsp)
      :                   double x3, x4, y3, y4;
      :                   x3 = x4 = 0;
      :                   y3 = y4 = 0;
      :
      :                   for(int h=0; h<2; ++h) {
      :                       double x1 = listLines[h]->at(k).first.x();
 0.00 :   40247d:       mov    (%rdx,%rbx,8),%rax
      :         public:
      :             virtual void raise() const;
      :             virtual Exception *clone() const;
      :         };
      :
      :         class Q_CORE_EXPORT UnhandledException : public Exception
 0.00 :   402481:       mov    (%rax),%rdx
      :
      :                 // [23.2.4.2] capacity
      :                 /**  Returns the number of elements in the %vector.  */
      :                 size_type
      :                 size() const _GLIBCXX_NOEXCEPT
      :                 { return size_type(this->_M_impl._M_finish - this->_M_impl._M_start); }
 0.00 :   402484:       mov    0x8(%rax),%rax
```

```
 0.00 :          402488:       sub    %rdx,%rax
 0.00 :          40248b:       sar    $0x4,%rax
      :            protected:
      :              /// Safety check used only from at().
      :              void
      :              _M_range_check(size_type __n) const
      :              {
      :                if (__n >= this->size())
 0.00 :          40248f:       cmp    %rbp,%rax
 0.00 :          402492:       jbe    4027f8 <Model::prepStraightLine()+0x3e8>
      :               *  out_of_range lookups are not defined. (For checked lookups
      :               *  see at().)
      :               */
      :              reference
      :              operator[](size_type __n)
      :              { return *(this->_M_impl._M_start + __n); }
 0.00 :          402498:       lea    (%rdx,%r13,1),%rax
 0.00 :          40249c:       cvtsi2sdl (%rax),%xmm1
      :                    double y1 = listLines[h]->at(k).first.y();
 0.00 :          4024a0:       cvtsi2sdl 0x4(%rax),%xmm0
      :
      :                    double x2 = listLines[h]->at(k).second.x();
 0.00 :          4024a5:       cvtsi2sdl 0x8(%rax),%xmm2
      :                    double y2 = listLines[h]->at(k).second.y();
 0.00 :          4024aa:       cvtsi2sdl 0xc(%rax),%xmm3
      :
      :               for(int h=0; h<2; ++h) {
      :                    double x1 = listLines[h]->at(k).first.x();
      :                    double y1 = listLines[h]->at(k).first.y();
      :
      :                    double x2 = listLines[h]->at(k).second.x();
 0.00 :          4024af:       movsd  %xmm2,0x18(%rsp)
      :                    double y2 = listLines[h]->at(k).second.y();
      :
      :                    double m = (y2 - y1) / (x2 - x1);
 0.00 :          4024b5:       subsd  %xmm1,%xmm2
      :               for(int h=0; h<2; ++h) {
      :                    double x1 = listLines[h]->at(k).first.x();
      :                    double y1 = listLines[h]->at(k).first.y();
      :
      :                    double x2 = listLines[h]->at(k).second.x();
      :                    double y2 = listLines[h]->at(k).second.y();
 0.00 :          4024b9:       movsd  %xmm3,0x20(%rsp)
      :
      :                    double m = (y2 - y1) / (x2 - x1);
 0.00 :          4024bf:       subsd  %xmm0,%xmm3
 0.00 :          4024c3:       divsd  %xmm2,%xmm3
      :
      :                    x3 += x1 / 2;
 0.00 :          4024c7:       movsd  0x441(%rip),%xmm2        # 402910 <_IO_stdin_used+0x60>
 0.00 :          4024cf:       mulsd  %xmm1,%xmm2
      :                    double y1 = listLines[h]->at(k).first.y();
      :
      :                    double x2 = listLines[h]->at(k).second.x();
      :                    double y2 = listLines[h]->at(k).second.y();
      :
      :                    double m = (y2 - y1) / (x2 - x1);
 0.00 :          4024d3:       movsd  %xmm3,0x28(%rsp)
      :
      :                    y3 += y1 / 2;
      :                    y4 += y2 / 2;
      :
      :                    // if x's are further apart than 'y's
      :                    if(fabs(x1-x2) > fabs(y1-y2)) {
```

```
 0.00 :         4024d9:       movapd %xmm1,%xmm3
 0.00 :         4024dd:       subsd  0x18(%rsp),%xmm3
      :                   double x2 = listLines[h]->at(k).second.x();
      :                   double y2 = listLines[h]->at(k).second.y();
      :
      :                   double m = (y2 - y1) / (x2 - x1);
      :
      :                   x3 += x1 / 2;
 0.00 :         4024e3:       addsd  0x38(%rsp),%xmm2
      :
      :                   y3 += y1 / 2;
      :                   y4 += y2 / 2;
      :
      :                   // if x's are further apart than 'y's
      :                   if(fabs(x1-x2) > fabs(y1-y2)) {
 0.00 :         4024e9:       movapd %xmm3,%xmm5
      :                   double x2 = listLines[h]->at(k).second.x();
      :                   double y2 = listLines[h]->at(k).second.y();
      :
      :                   double m = (y2 - y1) / (x2 - x1);
      :
      :                   x3 += x1 / 2;
 0.00 :         4024ed:       movsd  %xmm2,0x38(%rsp)
      :                   x4 += x2 / 2;
 0.00 :         4024f3:       movsd  0x18(%rsp),%xmm2
 0.00 :         4024f9:       mulsd  0x40f(%rip),%xmm2        # 402910 <_IO_stdin_used+0x60>
      :
      :                   y3 += y1 / 2;
      :                   y4 += y2 / 2;
      :
      :                   // if x's are further apart than 'y's
      :                   if(fabs(x1-x2) > fabs(y1-y2)) {
 0.00 :         402501:       andpd  0x427(%rip),%xmm5        # 402930 <_IO_stdin_used+0x80>
      :                   double y2 = listLines[h]->at(k).second.y();
      :
      :                   double m = (y2 - y1) / (x2 - x1);
      :
      :                   x3 += x1 / 2;
      :                   x4 += x2 / 2;
 0.00 :         402509:       addsd  0x40(%rsp),%xmm2
 0.00 :         40250f:       movsd  %xmm2,0x40(%rsp)
      :
      :                   y3 += y1 / 2;
 0.00 :         402515:       movsd  0x3f3(%rip),%xmm2        # 402910 <_IO_stdin_used+0x60>
 0.00 :         40251d:       mulsd  %xmm0,%xmm2
 0.00 :         402521:       addsd  0x48(%rsp),%xmm2
 0.00 :         402527:       movsd  %xmm2,0x48(%rsp)
      :                   y4 += y2 / 2;
 0.00 :         40252d:       movsd  0x20(%rsp),%xmm2
 0.00 :         402533:       mulsd  0x3d5(%rip),%xmm2        # 402910 <_IO_stdin_used+0x60>
 0.00 :         40253b:       addsd  0x50(%rsp),%xmm2
 0.00 :         402541:       movsd  %xmm2,0x50(%rsp)
      :
      :                   // if x's are further apart than 'y's
      :                   if(fabs(x1-x2) > fabs(y1-y2)) {
 0.00 :         402547:       movapd %xmm0,%xmm2
 0.00 :         40254b:       subsd  0x20(%rsp),%xmm2
 0.00 :         402551:       movapd %xmm2,%xmm4
 0.00 :         402555:       andpd  0x3d3(%rip),%xmm4        # 402930 <_IO_stdin_used+0x80>
 0.00 :         40255d:       ucomisd %xmm4,%xmm5
 0.00 :         402561:       jbe    4026d8 <Model::prepStraightLine()+0x2c8>
      :                        // swap coordinates if backwards
      :                        if(x1 > x2) { swap(x1, x2); swap(y1, y2); }
 0.00 :         402567:       ucomisd 0x18(%rsp),%xmm1
```

```
 0.00 :          40256d:       jbe    402788 <Model::prepStraightLine()+0x378>
      :              // concept requirements
      :              __glibcxx_function_requires(_SGIAssignableConcept<_Tp>)
      :
      :              _Tp __tmp = _GLIBCXX_MOVE(__a);
      :              __a = _GLIBCXX_MOVE(__b);
      :              __b = _GLIBCXX_MOVE(__tmp);
 0.00 :          402573:       movsd  %xmm0,0x20(%rsp)
 0.00 :          402579:       movapd %xmm1,%xmm0
      :            {
      :              // concept requirements
      :              __glibcxx_function_requires(_SGIAssignableConcept<_Tp>)
      :
      :              _Tp __tmp = _GLIBCXX_MOVE(__a);
      :              __a = _GLIBCXX_MOVE(__b);
 0.00 :          40257d:       movsd  0x18(%rsp),%xmm1
      :              __b = _GLIBCXX_MOVE(__tmp);
 0.00 :          402583:       movsd  %xmm0,0x18(%rsp)
 0.00 :          402589:       movapd %xmm1,%xmm3
 0.00 :          40258d:       subsd  %xmm0,%xmm3
 0.00 :          402591:       movsd  0x3b7(%rip),%xmm0        # 402950 <_IO_stdin_used+0xa0>
 0.00 :          402599:       movsd  %xmm0,0x30(%rsp)
      :
      :                        // draw line, with the appropriate slope
      :                        while(x1 < x2) {
      :                            y1 = m*(x1 - x2) + y2;
      :                            imgs[4]->setPixel(x1, y1, h?red:blue);
 0.00 :          40259f:       mov    %ebx,%r14d
 0.00 :          4025a2:       jmp    4025b0 <Model::prepStraightLine()+0x1a0>
 0.00 :          4025a4:       nopl   0x0(%rax)
 0.00 :          4025a8:       movapd %xmm1,%xmm3
 0.00 :          4025ac:       subsd  %xmm2,%xmm3
      :                        // swap coordinates if backwards
      :                        if(x1 > x2) { swap(x1, x2); swap(y1, y2); }
      :
      :                        // draw line, with the appropriate slope
      :                        while(x1 < x2) {
      :                            y1 = m*(x1 - x2) + y2;
 0.00 :          4025b0:       mulsd  0x28(%rsp),%xmm3
      :                            imgs[4]->setPixel(x1, y1, h?red:blue);
 0.00 :          4025b6:       cvttsd2si %xmm1,%esi
 0.00 :          4025ba:       cmp    $0x1,%r14d
 0.00 :          4025be:       mov    0x38(%r15),%rdi
 0.00 :          4025c2:       movsd  %xmm1,(%rsp)
 0.00 :          4025c7:       sbb    %ecx,%ecx
 0.00 :          4025c9:       and    $0xff0100ff,%ecx
 0.00 :          4025cf:       sub    $0x10000,%ecx
      :                        // swap coordinates if backwards
      :                        if(x1 > x2) { swap(x1, x2); swap(y1, y2); }
      :
      :                        // draw line, with the appropriate slope
      :                        while(x1 < x2) {
      :                            y1 = m*(x1 - x2) + y2;
 0.00 :          4025d5:       addsd  0x20(%rsp),%xmm3
      :                            imgs[4]->setPixel(x1, y1, h?red:blue);
 0.00 :          4025db:       cvttsd2si %xmm3,%edx
 7.41 :          4025df:       callq  400f20 <QImage::setPixel(int, int, unsigned int)@plt>
      :                            x1 += 0.01;
 3.70 :          4025e4:       movsd  (%rsp),%xmm1
      :                    if(fabs(x1-x2) > fabs(y1-y2)) {
      :                        // swap coordinates if backwards
      :                        if(x1 > x2) { swap(x1, x2); swap(y1, y2); }
      :
      :                        // draw line, with the appropriate slope
```

```
      :                         while(x1 < x2) {
 0.00 :         4025e9:       movsd  0x18(%rsp),%xmm2
      :                             y1 = m*(x1 - x2) + y2;
      :                             imgs[4]->setPixel(x1, y1, h?red:blue);
      :                             x1 += 0.01;
 0.00 :         4025ef:       addsd  0x30(%rsp),%xmm1
      :                   if(fabs(x1-x2) > fabs(y1-y2)) {
      :                         // swap coordinates if backwards
      :                         if(x1 > x2) { swap(x1, x2); swap(y1, y2); }
      :
      :                         // draw line, with the appropriate slope
      :                         while(x1 < x2) {
 3.70 :         4025f5:       ucomisd %xmm1,%xmm2
 0.00 :         4025f9:       ja     4025a8 <Model::prepStraightLine()+0x198>
 0.00 :         4025fb:       add    $0x1,%rbx
      :           for(unsigned int k=0; k<listLines[0]->size(); ++k) {
      :               double x3, x4, y3, y4;
      :               x3 = x4 = 0;
      :               y3 = y4 = 0;
      :
      :               for(int h=0; h<2; ++h) {
 0.00 :         4025ff:       cmp    $0x2,%rbx
 0.00 :         402603:       jne    402779 <Model::prepStraightLine()+0x369>
      :                             y1 += 0.01;
      :                         }
      :                     }
      :               }
      :
      :               pair<QPoint, QPoint> p = make_pair(QPoint(x3, y3), QPoint(x4, y4));
 0.00 :         402609:       movsd  0x38(%rsp),%xmm0
 0.00 :         40260f:       movsd  0x48(%rsp),%xmm1
 0.00 :         402615:       cvttsd2si %xmm0,%eax
 0.00 :         402619:       movsd  0x40(%rsp),%xmm2
 0.00 :         40261f:       movsd  0x50(%rsp),%xmm3
 0.00 :         402625:       mov    %eax,0x60(%rsp)
 0.00 :         402629:       cvttsd2si %xmm1,%eax
 0.00 :         40262d:       mov    %eax,0x64(%rsp)
 0.00 :         402631:       cvttsd2si %xmm2,%eax
 0.00 :         402635:       mov    %eax,0x68(%rsp)
 0.00 :         402639:       cvttsd2si %xmm3,%eax
 0.00 :         40263d:       mov    %eax,0x6c(%rsp)
      :               listAux[0]->push_back(p);
 0.00 :         402641:       mov    0x10(%r15),%rax
 0.00 :         402645:       mov    (%rax),%rdi
      :              *  available.
      :              */
      :             void
      :             push_back(const value_type& __x)
      :             {
      :               if (this->_M_impl._M_finish != this->_M_impl._M_end_of_storage)
 0.00 :         402648:       mov    0x8(%rdi),%rsi
 0.00 :         40264c:       cmp    0x10(%rdi),%rsi
 0.00 :         402650:       je     4027e5 <Model::prepStraightLine()+0x3d5>
      :       #else
      :             // _GLIBCXX_RESOLVE_LIB_DEFECTS
      :             // 402. wrong new expression in [some_] allocator::construct
      :             void
      :             construct(pointer __p, const _Tp& __val)
      :             { ::new((void *)__p) _Tp(__val); }
 0.00 :         402656:       test   %rsi,%rsi
 0.00 :         402659:       je     40266c <Model::prepStraightLine()+0x25c>
 0.00 :         40265b:       mov    0x60(%rsp),%rdx
 0.00 :         402660:       mov    %rdx,(%rsi)
 0.00 :         402663:       mov    0x68(%rsp),%rdx
```

```
 0.00 :          402668:       mov    %rdx,0x8(%rsi)
      :                  {
      :                    _Alloc_traits::construct(this->_M_impl, this->_M_impl._M_finish,
      :                                             __x);
      :                    ++this->_M_impl._M_finish;
 0.00 :          40266c:       add    $0x10,%rsi
 0.00 :          402670:       mov    %rsi,0x8(%rdi)
      :                listAux[1]->push_back(p);
 0.00 :          402674:       mov    0x8(%rax),%rdi
      :               *  available.
      :               */
      :              void
      :              push_back(const value_type& __x)
      :              {
      :                if (this->_M_impl._M_finish != this->_M_impl._M_end_of_storage)
 0.00 :          402678:       mov    0x8(%rdi),%rsi
 0.00 :          40267c:       cmp    0x10(%rdi),%rsi
 0.00 :          402680:       je     4027d3 <Model::prepStraightLine()+0x3c3>
 0.00 :          402686:       test   %rsi,%rsi
 0.00 :          402689:       je     40269c <Model::prepStraightLine()+0x28c>
 0.00 :          40268b:       mov    0x60(%rsp),%rax
 0.00 :          402690:       mov    %rax,(%rsi)
 0.00 :          402693:       mov    0x68(%rsp),%rax
 0.00 :          402698:       mov    %rax,0x8(%rsi)
      :                  {
      :                    _Alloc_traits::construct(this->_M_impl, this->_M_impl._M_finish,
      :                                             __x);
      :                    ++this->_M_impl._M_finish;
 0.00 :          40269c:       add    $0x10,%rsi
 0.00 :          4026a0:       mov    %rsi,0x8(%rdi)
      :
      :          void Model::prepStraightLine() {
      :              QRgb red = qRgb(255, 0, 0);
      :              QRgb blue = qRgb(0, 0, 255);
      :
      :              for(unsigned int k=0; k<listLines[0]->size(); ++k) {
 0.00 :          4026a4:       mov    0x8(%r15),%rdx
 0.00 :          4026a8:       add    $0x1,%r12d
 0.00 :          4026ac:       mov    %r12d,%ebp
 0.00 :          4026af:       mov    (%rdx),%rcx
      :
      :                  // [23.2.4.2] capacity
      :                  /**  Returns the number of elements in the %vector.  */
      :                  size_type
      :                  size() const _GLIBCXX_NOEXCEPT
      :                  { return size_type(this->_M_impl._M_finish - this->_M_impl._M_start); }
 0.00 :          4026b2:       mov    0x8(%rcx),%rax
 0.00 :          4026b6:       sub    (%rcx),%rax
 0.00 :          4026b9:       sar    $0x4,%rax
 0.00 :          4026bd:       cmp    %rax,%rbp
 0.00 :          4026c0:       jb     402450 <Model::prepStraightLine()+0x40>
      :
      :                pair<QPoint, QPoint> p = make_pair(QPoint(x3, y3), QPoint(x4, y4));
      :                listAux[0]->push_back(p);
      :                listAux[1]->push_back(p);
      :              }
      :          }
 0.00 :          4026c6:       add    $0x78,%rsp
 0.00 :          4026ca:       pop    %rbx
 0.00 :          4026cb:       pop    %rbp
 0.00 :          4026cc:       pop    %r12
 0.00 :          4026ce:       pop    %r13
 0.00 :          4026d0:       pop    %r14
 0.00 :          4026d2:       pop    %r15
```

```
  0.00 :          4026d4:       retq
  0.00 :          4026d5:       nopl   (%rax)
       :                            imgs[4]->setPixel(x1, y1, h?red:blue);
       :                            x1 += 0.01;
       :                        }
       :                    } else {
       :                        // swap coordinates if backwards
       :                        if(y1 > y2) { swap(x1, x2); swap(y1, y2); }
  0.00 :          4026d8:       ucomisd 0x20(%rsp),%xmm0
  0.00 :          4026de:       jbe    4027b0 <Model::prepStraightLine()+0x3a0>
  0.00 :          4026e4:       movsd  %xmm1,0x18(%rsp)
  0.00 :          4026ea:       movapd %xmm0,%xmm1
       :            {
       :              // concept requirements
       :              __glibcxx_function_requires(_SGIAssignableConcept<_Tp>)
       :
       :              _Tp __tmp = _GLIBCXX_MOVE(__a);
       :              __a = _GLIBCXX_MOVE(__b);
  0.00 :          4026ee:       movsd  0x20(%rsp),%xmm0
  0.00 :          4026f4:       movsd  0x58(%rsp),%xmm3
  0.00 :          4026fa:       movapd %xmm0,%xmm2
       :              __b = _GLIBCXX_MOVE(__tmp);
  0.00 :          4026fe:       movsd  %xmm1,0x20(%rsp)
  0.00 :          402704:       movsd  %xmm3,0x30(%rsp)
  0.00 :          40270a:       subsd  %xmm1,%xmm2
       :
       :                        // draw line, with the appropriate slope
       :                        while(y1 < y2) {
       :                            x1 = (y1 - y2)/m + x2;
       :                            imgs[4]->setPixel(x1, y1, h?red:blue);
  0.00 :          40270e:       mov    %ebx,%r14d
  0.00 :          402711:       jmp    402720 <Model::prepStraightLine()+0x310>
  0.00 :          402713:       nopl   0x0(%rax,%rax,1)
  3.70 :          402718:       movapd %xmm0,%xmm2
  0.00 :          40271c:       subsd  %xmm1,%xmm2
       :                        // swap coordinates if backwards
       :                        if(y1 > y2) { swap(x1, x2); swap(y1, y2); }
       :
       :                        // draw line, with the appropriate slope
       :                        while(y1 < y2) {
       :                            x1 = (y1 - y2)/m + x2;
  0.00 :          402720:       divsd  0x28(%rsp),%xmm2
       :                            imgs[4]->setPixel(x1, y1, h?red:blue);
 51.85 :          402726:       cvttsd2si %xmm0,%edx
  0.00 :          40272a:       cmp    $0x1,%r14d
  0.00 :          40272e:       mov    0x38(%r15),%rdi
  0.00 :          402732:       movsd  %xmm0,(%rsp)
  0.00 :          402737:       sbb    %ecx,%ecx
  0.00 :          402739:       and    $0xff0100ff,%ecx
  0.00 :          40273f:       sub    $0x10000,%ecx
       :                        // swap coordinates if backwards
       :                        if(y1 > y2) { swap(x1, x2); swap(y1, y2); }
       :
       :                        // draw line, with the appropriate slope
       :                        while(y1 < y2) {
       :                            x1 = (y1 - y2)/m + x2;
  0.00 :          402745:       addsd  0x18(%rsp),%xmm2
       :                            imgs[4]->setPixel(x1, y1, h?red:blue);
  0.00 :          40274b:       cvttsd2si %xmm2,%esi
 22.22 :          40274f:       callq  400f20 <QImage::setPixel(int, int, unsigned int)@plt>
       :                            y1 += 0.01;
  0.00 :          402754:       movsd  (%rsp),%xmm0
       :                    } else {
       :                        // swap coordinates if backwards
```

```
         :                          if(y1 > y2) { swap(x1, x2); swap(y1, y2); }
         :
         :                          // draw line, with the appropriate slope
         :                          while(y1 < y2) {
    3.70 :          402759:         movsd  0x20(%rsp),%xmm1
         :                              x1 = (y1 - y2)/m + x2;
         :                              imgs[4]->setPixel(x1, y1, h?red:blue);
         :                              y1 += 0.01;
    0.00 :          40275f:         addsd  0x30(%rsp),%xmm0
         :                      } else {
         :                          // swap coordinates if backwards
         :                          if(y1 > y2) { swap(x1, x2); swap(y1, y2); }
         :
         :                          // draw line, with the appropriate slope
         :                          while(y1 < y2) {
    0.00 :          402765:         ucomisd %xmm0,%xmm1
    3.70 :          402769:         ja     402718 <Model::prepStraightLine()+0x308>
    0.00 :          40276b:         add    $0x1,%rbx
         :            for(unsigned int k=0; k<listLines[0]->size(); ++k) {
         :                double x3, x4, y3, y4;
         :                x3 = x4 = 0;
         :                y3 = y4 = 0;
         :
         :                for(int h=0; h<2; ++h) {
    0.00 :          40276f:         cmp    $0x2,%rbx
    0.00 :          402773:         je     402609 <Model::prepStraightLine()+0x1f9>
    0.00 :          402779:         mov    0x8(%r15),%rdx
    0.00 :          40277d:         jmpq   40247d <Model::prepStraightLine()+0x6d>
    0.00 :          402782:         nopw   0x0(%rax,%rax,1)
         :                       if(fabs(x1-x2) > fabs(y1-y2)) {
         :                          // swap coordinates if backwards
         :                          if(x1 > x2) { swap(x1, x2); swap(y1, y2); }
         :
         :                          // draw line, with the appropriate slope
         :                          while(x1 < x2) {
    0.00 :          402788:         movsd  0x18(%rsp),%xmm0
    0.00 :          40278e:         movsd  0x1ba(%rip),%xmm2        # 402950 <_IO_stdin_used+0xa0>
    0.00 :          402796:         ucomisd %xmm1,%xmm0
    0.00 :          40279a:         movsd  %xmm2,0x30(%rsp)
    0.00 :          4027a0:         ja     40259f <Model::prepStraightLine()+0x18f>
    0.00 :          4027a6:         jmpq   4025fb <Model::prepStraightLine()+0x1eb>
    0.00 :          4027ab:         nopl   0x0(%rax,%rax,1)
         :                       } else {
         :                          // swap coordinates if backwards
         :                          if(y1 > y2) { swap(x1, x2); swap(y1, y2); }
         :
         :                          // draw line, with the appropriate slope
         :                          while(y1 < y2) {
    0.00 :          4027b0:         movsd  0x20(%rsp),%xmm3
    0.00 :          4027b6:         movsd  0x192(%rip),%xmm1        # 402950 <_IO_stdin_used+0xa0>
    0.00 :          4027be:         ucomisd %xmm0,%xmm3
    0.00 :          4027c2:         movsd  %xmm1,0x30(%rsp)
    0.00 :          4027c8:         ja     40270e <Model::prepStraightLine()+0x2fe>
    0.00 :          4027ce:         jmpq   4025fb <Model::prepStraightLine()+0x1eb>
         :                      }
         :                     else
         :           #ifdef __GXX_EXPERIMENTAL_CXX0X__
         :                       _M_emplace_back_aux(__x);
         :           #else
         :                       _M_insert_aux(end(), __x);
    0.00 :            4027d3:       lea    0x60(%rsp),%rdx
    0.00 :            4027d8:       callq  401690 <std::vector<std::pair<QPoint, QPoint>, std::allocator<std::pair<QPoint,
QPoint> > >::_M_insert_aux(__gnu_cxx::__normal_iterator<std::pair<QPoint, QPoint>*, std::vector<std::pair<QPoint, QPoint>,
std::allocator<std::pair<QPoint, QPoint> > > >, std::pair<QPoint, QPoint> const&)>
```

```
    0.00 :          4027dd:       nopl   (%rax)
    0.00 :          4027e0:       jmpq   4026a4 <Model::prepStraightLine()+0x294>
    0.00 :          4027e5:       lea    0x60(%rsp),%rdx
    0.00 :          4027ea:       callq  401690 <std::vector<std::pair<QPoint, QPoint>, std::allocator<std::pair<QPoint,
QPoint> > >::_M_insert_aux(__gnu_cxx::__normal_iterator<std::pair<QPoint, QPoint>*, std::vector<std::pair<QPoint, QPoint>,
std::allocator<std::pair<QPoint, QPoint> > > >, std::pair<QPoint, QPoint> const&)>
    0.00 :          4027ef:       mov    0x10(%r15),%rax
    0.00 :          4027f3:       jmpq   402674 <Model::prepStraightLine()+0x264>
         :              /// Safety check used only from at().
         :              void
         :              _M_range_check(size_type __n) const
         :              {
         :                if (__n >= this->size())
         :                  __throw_out_of_range(__N("vector::_M_range_check"));
    0.00 :          4027f8:       mov    $0x402918,%edi
    0.00 :          4027fd:       callq  400f10 <std::__throw_out_of_range(char const*)@plt>
 Percent |       Source code & Disassembly of test_harness
-------------------------------------------------
         :
         :
         :
         :       Disassembly of section .plt:
         :
         :       0000000000400f20 <QImage::setPixel(int, int, unsigned int)@plt>:
  100.00 :          400f20:       jmpq   *0x202192(%rip)        # 6030b8 <_GLOBAL_OFFSET_TABLE_+0xb8>
    0.00 :          400f26:       pushq  $0x14
    0.00 :          400f2b:       jmpq   400dd0 <_init+0x20>
 Percent |       Source code & Disassembly of test_harness
-------------------------------------------------
         :
         :
         :
         :       Disassembly of section .plt:
         :
         :       0000000000400f40 <QVector2D::lengthSquared() const@plt>:
  100.00 :          400f40:       jmpq   *0x202182(%rip)        # 6030c8 <_GLOBAL_OFFSET_TABLE_+0xc8>
    0.00 :          400f46:       pushq  $0x16
    0.00 :          400f4b:       jmpq   400dd0 <_init+0x20>
 Percent |       Source code & Disassembly of test_harness
-------------------------------------------------
         :
         :
         :
         :       Disassembly of section .text:
         :
         :       0000000000401950 <Model::commonPrep()>:
         :              listAux[0]->push_back(p);
         :              listAux[1]->push_back(p);
         :          }
         :       }
         :
         :       void Model::commonPrep() {
    0.00 :          401950:       push   %rbp
    0.00 :          401951:       push   %rbx
    0.00 :          401952:       mov    %rdi,%rbx
    0.00 :          401955:       sub    $0x38,%rsp
         :            for(unsigned int k=0; k<listAux[0]->size(); ++k) {
    0.00 :          401959:       mov    0x10(%rdi),%rdi
    0.00 :          40195d:       mov    (%rdi),%rdx
         :       public:
         :           virtual void raise() const;
         :           virtual Exception *clone() const;
         :       };
         :
```

```
        :        class Q_CORE_EXPORT UnhandledException : public Exception
   0.00 :          401960:       mov    (%rdx),%rax
        :
        :            // [23.2.4.2] capacity
        :            /**  Returns the number of elements in the %vector.  */
        :            size_type
        :            size() const _GLIBCXX_NOEXCEPT
        :            { return size_type(this->_M_impl._M_finish - this->_M_impl._M_start); }
   0.00 :          401963:       mov    0x8(%rdx),%rdx
   0.00 :          401967:       sub    %rax,%rdx
   0.00 :          40196a:       sar    $0x4,%rdx
   0.00 :          40196e:       test   %rdx,%rdx
   0.00 :          401971:       je     401b78 <Model::commonPrep()+0x228>
        :                double x3, x4, y3, y4;
        :
        :                x3 = listAux[0]->at(k).first.x();
        :                y3 = listAux[0]->at(k).first.y();
        :
        :                x4 = listAux[0]->at(k).second.x();
   0.00 :          401977:       cvtsi2sdl 0x8(%rax),%xmm0
   0.00 :          40197c:       movsd  %xmm0,0x18(%rsp)
   0.00 :          401982:       xor    %ebp,%ebp
        :
        :        void Model::commonPrep() {
        :            for(unsigned int k=0; k<listAux[0]->size(); ++k) {
        :                double x3, x4, y3, y4;
        :
        :                x3 = listAux[0]->at(k).first.x();
   0.00 :          401984:       cvtsi2sdl (%rax),%xmm2
        :                y3 = listAux[0]->at(k).first.y();
   0.00 :          401988:       cvtsi2sdl 0x4(%rax),%xmm1
   0.00 :          40198d:       nopl   (%rax)
        :
        :                x4 = listAux[0]->at(k).second.x();
        :                y4 = listAux[0]->at(k).second.y();
   0.00 :          401990:       cvtsi2sdl 0xc(%rax),%xmm0
        :
        :                double m = (y4 - y3) / (x4 - x3);
   0.00 :          401995:       movsd  0x18(%rsp),%xmm3
        :
        :                x3 = listAux[0]->at(k).first.x();
        :                y3 = listAux[0]->at(k).first.y();
        :
        :                x4 = listAux[0]->at(k).second.x();
        :                y4 = listAux[0]->at(k).second.y();
   0.00 :          40199b:       movsd  %xmm0,0x10(%rsp)
        :
        :                double m = (y4 - y3) / (x4 - x3);
   0.00 :          4019a1:       subsd  %xmm1,%xmm0
   0.00 :          4019a5:       subsd  %xmm2,%xmm3
   0.00 :          4019a9:       movapd %xmm0,%xmm4
        :                // if x's are further apart than 'y's
        :                if(fabs(x4-x3) > fabs(y4-y3)) {
   0.00 :          4019ad:       andpd  0xf7b(%rip),%xmm0        # 402930 <_IO_stdin_used+0x80>
        :                y3 = listAux[0]->at(k).first.y();
        :
        :                x4 = listAux[0]->at(k).second.x();
        :                y4 = listAux[0]->at(k).second.y();
        :
        :                double m = (y4 - y3) / (x4 - x3);
   0.00 :          4019b5:       divsd  %xmm3,%xmm4
        :                // if x's are further apart than 'y's
        :                if(fabs(x4-x3) > fabs(y4-y3)) {
   0.00 :          4019b9:       andpd  0xf6f(%rip),%xmm3        # 402930 <_IO_stdin_used+0x80>
```

```
        :                y3 = listAux[0]->at(k).first.y();
        :
        :                x4 = listAux[0]->at(k).second.x();
        :                y4 = listAux[0]->at(k).second.y();
        :
        :                double m = (y4 - y3) / (x4 - x3);
   0.00 :          4019c1:       movsd  %xmm4,0x20(%rsp)
        :                // if x's are further apart than 'y's
        :                if(fabs(x4-x3) > fabs(y4-y3)) {
   0.00 :          4019c7:       ucomisd %xmm0,%xmm3
   0.00 :          4019cb:       jbe    401aa0 <Model::commonPrep()+0x150>
        :                    // swap coordinates if backwards
        :                    if(x3 > x4) { swap(x3, x4); swap(y3, y4); }
   0.00 :          4019d1:       ucomisd 0x18(%rsp),%xmm2
   0.00 :          4019d7:       jbe    401b28 <Model::commonPrep()+0x1d8>
   0.00 :          4019dd:       movapd %xmm2,%xmm0
   0.00 :          4019e1:       movabs $0x3f847ae147ae147b,%rsi
   0.00 :          4019eb:       movsd  0x18(%rsp),%xmm2
   0.00 :          4019f1:       mov    %rsi,0x28(%rsp)
   0.00 :          4019f6:       movsd  %xmm1,0x10(%rsp)
   0.00 :          4019fc:       movsd  %xmm0,0x18(%rsp)
   0.00 :          401a02:       nopw   0x0(%rax,%rax,1)
        :
        :                    // draw line, with the appropriate slope
        :                    while(x3 < x4) {
        :                        y3 = m*(x3 - x4) + y4;
   0.00 :          401a08:       movapd %xmm2,%xmm0
        :                        imgs[4]->setPixel(x3, y3, qRgb(0, 0, 0));
   0.00 :          401a0c:       mov    0x38(%rbx),%rdi
   0.00 :          401a10:       cvttsd2si %xmm2,%esi
   0.00 :          401a14:       mov    $0xff000000,%ecx
        :                    // swap coordinates if backwards
        :                    if(x3 > x4) { swap(x3, x4); swap(y3, y4); }
        :
        :                    // draw line, with the appropriate slope
        :                    while(x3 < x4) {
        :                        y3 = m*(x3 - x4) + y4;
   0.00 :          401a19:       subsd  0x18(%rsp),%xmm0
        :                        imgs[4]->setPixel(x3, y3, qRgb(0, 0, 0));
   0.00 :          401a1f:       movsd  %xmm2,(%rsp)
        :                    // swap coordinates if backwards
        :                    if(x3 > x4) { swap(x3, x4); swap(y3, y4); }
        :
        :                    // draw line, with the appropriate slope
        :                    while(x3 < x4) {
        :                        y3 = m*(x3 - x4) + y4;
   0.00 :          401a24:       mulsd  0x20(%rsp),%xmm0
   0.00 :          401a2a:       addsd  0x10(%rsp),%xmm0
        :                        imgs[4]->setPixel(x3, y3, qRgb(0, 0, 0));
  12.50 :          401a30:       cvttsd2si %xmm0,%edx
   0.00 :          401a34:       callq  400f20 <QImage::setPixel(int, int, unsigned int)@plt>
        :                        x3 += 0.01;
   0.00 :          401a39:       movsd  (%rsp),%xmm2
        :                if(fabs(x4-x3) > fabs(y4-y3)) {
        :                    // swap coordinates if backwards
        :                    if(x3 > x4) { swap(x3, x4); swap(y3, y4); }
        :
        :                    // draw line, with the appropriate slope
        :                    while(x3 < x4) {
   0.00 :          401a3e:       movsd  0x18(%rsp),%xmm4
        :                        y3 = m*(x3 - x4) + y4;
        :                        imgs[4]->setPixel(x3, y3, qRgb(0, 0, 0));
        :                        x3 += 0.01;
   0.00 :          401a44:       addsd  0x28(%rsp),%xmm2
```

```
        :                if(fabs(x4-x3) > fabs(y4-y3)) {
        :                    // swap coordinates if backwards
        :                    if(x3 > x4) { swap(x3, x4); swap(y3, y4); }
        :
        :                    // draw line, with the appropriate slope
        :                    while(x3 < x4) {
   0.00 :          401a4a:       ucomisd %xmm2,%xmm4
   0.00 :          401a4e:       ja     401a08 <Model::commonPrep()+0xb8>
   0.00 :          401a50:       mov    0x10(%rbx),%rdi
        :                listAux[1]->push_back(p);
        :            }
        :        }
        :
        :        void Model::commonPrep() {
        :            for(unsigned int k=0; k<listAux[0]->size(); ++k) {
   0.00 :          401a54:       mov    (%rdi),%rdx
   0.00 :          401a57:       add    $0x1,%ebp
   0.00 :          401a5a:       mov    %ebp,%ecx
   0.00 :          401a5c:       mov    (%rdx),%rsi
   0.00 :          401a5f:       mov    0x8(%rdx),%rax
   0.00 :          401a63:       sub    %rsi,%rax
   0.00 :          401a66:       sar    $0x4,%rax
   0.00 :          401a6a:       cmp    %rax,%rcx
   0.00 :          401a6d:       jae    401b78 <Model::commonPrep()+0x228>
        :             *  out_of_range lookups are not defined. (For checked lookups
        :             *  see at().)
        :             */
        :            reference
        :            operator[](size_type __n)
        :            { return *(this->_M_impl._M_start + __n); }
   0.00 :          401a73:       mov    %rcx,%rax
   0.00 :          401a76:       shl    $0x4,%rax
   0.00 :          401a7a:       add    %rsi,%rax
        :                double x3, x4, y3, y4;
        :
        :                x3 = listAux[0]->at(k).first.x();
        :                y3 = listAux[0]->at(k).first.y();
        :
        :                x4 = listAux[0]->at(k).second.x();
   0.00 :          401a7d:       cvtsi2sdl 0x8(%rax),%xmm4
        :
        :        void Model::commonPrep() {
        :            for(unsigned int k=0; k<listAux[0]->size(); ++k) {
        :                double x3, x4, y3, y4;
        :
        :                x3 = listAux[0]->at(k).first.x();
   0.00 :          401a82:       cvtsi2sdl (%rax),%xmm2
        :                y3 = listAux[0]->at(k).first.y();
   0.00 :          401a86:       cvtsi2sdl 0x4(%rax),%xmm1
        :
        :                x4 = listAux[0]->at(k).second.x();
   0.00 :          401a8b:       movsd  %xmm4,0x18(%rsp)
   0.00 :          401a91:       jmpq   401990 <Model::commonPrep()+0x40>
   0.00 :          401a96:       nopw   %cs:0x0(%rax,%rax,1)
        :                    imgs[4]->setPixel(x3, y3, qRgb(0, 0, 0));
        :                    x3 += 0.01;
        :                }
        :            } else {
        :                // swap coordinates if backwards
        :                if(y3 > y4) { swap(x3, x4); swap(y3, y4); }
   0.00 :          401aa0:       ucomisd 0x10(%rsp),%xmm1
   0.00 :          401aa6:       jbe    401b50 <Model::commonPrep()+0x200>
   0.00 :          401aac:       movapd %xmm1,%xmm0
   0.00 :          401ab0:       movabs $0x3f847ae147ae147b,%rdx
```

```
        :            {
        :              // concept requirements
        :              __glibcxx_function_requires(_SGIAssignableConcept<_Tp>)
        :
        :              _Tp __tmp = _GLIBCXX_MOVE(__a);
        :              __a = _GLIBCXX_MOVE(__b);
   0.00 :       401aba:       movsd  0x10(%rsp),%xmm1
   0.00 :       401ac0:       mov    %rdx,0x28(%rsp)
   0.00 :       401ac5:       movsd  %xmm2,0x18(%rsp)
   0.00 :       401acb:       movsd  %xmm0,0x10(%rsp)
   0.00 :       401ad1:       nopl   0x0(%rax)
        :
        :                  // draw line, with the appropriate slope
        :                  while(y3 < y4) {
        :                      x3 = (y3 - y4)/m + x4;
   0.00 :       401ad8:       movapd %xmm1,%xmm0
        :                      imgs[4]->setPixel(x3, y3, qRgb(0, 0, 0));
   0.00 :       401adc:       mov    0x38(%rbx),%rdi
   0.00 :       401ae0:       cvttsd2si %xmm1,%edx
  12.50 :       401ae4:       mov    $0xff000000,%ecx
        :                  // swap coordinates if backwards
        :                  if(y3 > y4) { swap(x3, x4); swap(y3, y4); }
        :
        :                  // draw line, with the appropriate slope
        :                  while(y3 < y4) {
        :                      x3 = (y3 - y4)/m + x4;
   0.00 :       401ae9:       subsd  0x10(%rsp),%xmm0
        :                      imgs[4]->setPixel(x3, y3, qRgb(0, 0, 0));
   0.00 :       401aef:       movsd  %xmm1,(%rsp)
        :                  // swap coordinates if backwards
        :                  if(y3 > y4) { swap(x3, x4); swap(y3, y4); }
        :
        :                  // draw line, with the appropriate slope
        :                  while(y3 < y4) {
        :                      x3 = (y3 - y4)/m + x4;
   0.00 :       401af4:       divsd  0x20(%rsp),%xmm0
  25.00 :       401afa:       addsd  0x18(%rsp),%xmm0
        :                      imgs[4]->setPixel(x3, y3, qRgb(0, 0, 0));
  25.00 :       401b00:       cvttsd2si %xmm0,%esi
   0.00 :       401b04:       callq  400f20 <QImage::setPixel(int, int, unsigned int)@plt>
        :                      y3 += 0.01;
   0.00 :       401b09:       movsd  (%rsp),%xmm1
        :              } else {
        :                  // swap coordinates if backwards
        :                  if(y3 > y4) { swap(x3, x4); swap(y3, y4); }
        :
        :                  // draw line, with the appropriate slope
        :                  while(y3 < y4) {
  25.00 :       401b0e:       movsd  0x10(%rsp),%xmm4
        :                      x3 = (y3 - y4)/m + x4;
        :                      imgs[4]->setPixel(x3, y3, qRgb(0, 0, 0));
        :                      y3 += 0.01;
   0.00 :       401b14:       addsd  0x28(%rsp),%xmm1
        :              } else {
        :                  // swap coordinates if backwards
        :                  if(y3 > y4) { swap(x3, x4); swap(y3, y4); }
        :
        :                  // draw line, with the appropriate slope
        :                  while(y3 < y4) {
   0.00 :       401b1a:       ucomisd %xmm1,%xmm4
   0.00 :       401b1e:       ja     401ad8 <Model::commonPrep()+0x188>
   0.00 :       401b20:       jmpq   401a50 <Model::commonPrep()+0x100>
   0.00 :       401b25:       nopl   (%rax)
        :              if(fabs(x4-x3) > fabs(y4-y3)) {
```

```
         :                    // swap coordinates if backwards
         :                    if(x3 > x4) { swap(x3, x4); swap(y3, y4); }
         :
         :                    // draw line, with the appropriate slope
         :                    while(x3 < x4) {
    0.00 :          401b28:       movsd  0x18(%rsp),%xmm0
    0.00 :          401b2e:       movabs $0x3f847ae147ae147b,%rcx
    0.00 :          401b38:       mov    %rcx,0x28(%rsp)
    0.00 :          401b3d:       ucomisd %xmm2,%xmm0
    0.00 :          401b41:       ja     401a08 <Model::commonPrep()+0xb8>
    0.00 :          401b47:       jmpq   401a54 <Model::commonPrep()+0x104>
    0.00 :          401b4c:       nopl   0x0(%rax)
         :                } else {
         :                    // swap coordinates if backwards
         :                    if(y3 > y4) { swap(x3, x4); swap(y3, y4); }
         :
         :                    // draw line, with the appropriate slope
         :                    while(y3 < y4) {
    0.00 :          401b50:       movsd  0x10(%rsp),%xmm0
    0.00 :          401b56:       movabs $0x3f847ae147ae147b,%rax
    0.00 :          401b60:       mov    %rax,0x28(%rsp)
    0.00 :          401b65:       ucomisd %xmm1,%xmm0
    0.00 :          401b69:       ja     401ad8 <Model::commonPrep()+0x188>
    0.00 :          401b6f:       jmpq   401a54 <Model::commonPrep()+0x104>
    0.00 :          401b74:       nopl   0x0(%rax)
         :                        imgs[4]->setPixel(x3, y3, qRgb(0, 0, 0));
         :                        y3 += 0.01;
         :                    }
         :                }
         :            }
         :        }
    0.00 :          401b78:       add    $0x38,%rsp
    0.00 :          401b7c:       pop    %rbx
    0.00 :          401b7d:       pop    %rbp
    0.00 :          401b7e:       retq
 Percent |       Source code & Disassembly of test_harness
-------------------------------------------------
         :
         :
         :
         :       Disassembly of section .plt:
         :
         :       0000000000400ed0 <QVector2D::dotProduct(QVector2D const&, QVector2D const&)@plt>:
  100.00 :          400ed0:       jmpq   *0x2021ba(%rip)        # 603090 <_GLOBAL_OFFSET_TABLE_+0x90>
    0.00 :          400ed6:       pushq  $0xf
    0.00 :          400edb:       jmpq   400dd0 <_init+0x20>
 Percent |       Source code & Disassembly of libjpeg.so.8.0.2
-------------------------------------------------
         :
         :
         :
         :       Disassembly of section .text:
         :
         :       0000000000019d80 <jpeg_fill_bit_buffer>:
    0.00 :          19d80:       push   %r15
    0.00 :          19d82:       mov    %edx,%r15d
    0.00 :          19d85:       push   %r14
   25.00 :          19d87:       push   %r13
    0.00 :          19d89:       mov    %rdi,%r13
    0.00 :          19d8c:       push   %r12
    0.00 :          19d8e:       mov    %rsi,%r12
    0.00 :          19d91:       push   %rbp
    0.00 :          19d92:       push   %rbx
    0.00 :          19d93:       sub    $0x18,%rsp
```

```
 0.00 :        19d97:       mov    0x20(%rdi),%r14
 0.00 :        19d9b:       mov    (%rdi),%rbp
 0.00 :        19d9e:       mov    %ecx,0xc(%rsp)
 0.00 :        19da2:       mov    0x8(%rdi),%rbx
 0.00 :        19da6:       mov    0x234(%r14),%r11d
 0.00 :        19dad:       test   %r11d,%r11d
 0.00 :        19db0:       jne    19e78 <jpeg_fill_bit_buffer+0xf8>
 0.00 :        19db6:       cmp    $0x38,%edx
 0.00 :        19db9:       jle    19e25 <jpeg_fill_bit_buffer+0xa5>
 0.00 :        19dbb:       nopl   0x0(%rax,%rax,1)
 0.00 :        19dc0:       mov    %rbp,0x0(%r13)
 0.00 :        19dc4:       mov    %rbx,0x8(%r13)
 0.00 :        19dc8:       mov    $0x1,%eax
 0.00 :        19dcd:       mov    %r12,0x10(%r13)
 0.00 :        19dd1:       mov    %r15d,0x18(%r13)
 0.00 :        19dd5:       add    $0x18,%rsp
 0.00 :        19dd9:       pop    %rbx
 0.00 :        19dda:       pop    %rbp
 0.00 :        19ddb:       pop    %r12
 0.00 :        19ddd:       pop    %r13
 0.00 :        19ddf:       pop    %r14
 0.00 :        19de1:       pop    %r15
 0.00 :        19de3:       retq
 0.00 :        19de4:       mov    0x28(%r14),%rax
 0.00 :        19de8:       mov    %r14,%rdi
 0.00 :        19deb:       callq  *0x18(%rax)
 0.00 :        19dee:       test   %eax,%eax
 0.00 :        19df0:       je     19eb0 <jpeg_fill_bit_buffer+0x130>
 0.00 :        19df6:       mov    0x28(%r14),%rax
 0.00 :        19dfa:       mov    (%rax),%rbp
 0.00 :        19dfd:       mov    0x8(%rax),%rbx
25.00 :        19e01:       movzbl 0x0(%rbp),%eax
 0.00 :        19e05:       sub    $0x1,%rbx
 0.00 :        19e09:       add    $0x1,%rbp
 0.00 :        19e0d:       cmp    $0xff,%al
25.00 :        19e0f:       movzbl %al,%ecx
 0.00 :        19e12:       je     19e58 <jpeg_fill_bit_buffer+0xd8>
 0.00 :        19e14:       shl    $0x8,%r12
 0.00 :        19e18:       add    $0x8,%r15d
 0.00 :        19e1c:       or     %rcx,%r12
25.00 :        19e1f:       cmp    $0x38,%r15d
 0.00 :        19e23:       jg     19dc0 <jpeg_fill_bit_buffer+0x40>
 0.00 :        19e25:       test   %rbx,%rbx
 0.00 :        19e28:       jne    19e01 <jpeg_fill_bit_buffer+0x81>
 0.00 :        19e2a:       jmp    19de4 <jpeg_fill_bit_buffer+0x64>
 0.00 :        19e2c:       mov    0x28(%r14),%rax
 0.00 :        19e30:       mov    %r14,%rdi
 0.00 :        19e33:       callq  *0x18(%rax)
 0.00 :        19e36:       test   %eax,%eax
 0.00 :        19e38:       je     19eb0 <jpeg_fill_bit_buffer+0x130>
 0.00 :        19e3a:       mov    0x28(%r14),%rax
 0.00 :        19e3e:       mov    (%rax),%rbp
 0.00 :        19e41:       mov    0x8(%rax),%rbx
 0.00 :        19e45:       movzbl 0x0(%rbp),%eax
 0.00 :        19e49:       sub    $0x1,%rbx
 0.00 :        19e4d:       add    $0x1,%rbp
 0.00 :        19e51:       cmp    $0xff,%eax
 0.00 :        19e56:       jne    19e60 <jpeg_fill_bit_buffer+0xe0>
 0.00 :        19e58:       test   %rbx,%rbx
 0.00 :        19e5b:       jne    19e45 <jpeg_fill_bit_buffer+0xc5>
 0.00 :        19e5d:       jmp    19e2c <jpeg_fill_bit_buffer+0xac>
 0.00 :        19e5f:       nop
 0.00 :        19e60:       test   %eax,%eax
 0.00 :        19e62:       jne    19e6b <jpeg_fill_bit_buffer+0xeb>
```

```
    0.00 :          19e64:       mov    $0xff,%ecx
    0.00 :          19e69:       jmp    19e14 <jpeg_fill_bit_buffer+0x94>
    0.00 :          19e6b:       mov    %eax,0x234(%r14)
    0.00 :          19e72:       nopw   0x0(%rax,%rax,1)
    0.00 :          19e78:       cmp    0xc(%rsp),%r15d
    0.00 :          19e7d:       jge    19dc0 <jpeg_fill_bit_buffer+0x40>
    0.00 :          19e83:       mov    0x268(%r14),%rax
    0.00 :          19e8a:       mov    0x10(%rax),%r10d
    0.00 :          19e8e:       test   %r10d,%r10d
    0.00 :          19e91:       je     19ec1 <jpeg_fill_bit_buffer+0x141>
    0.00 :          19e93:       mov    $0x39,%ecx
    0.00 :          19e98:       sub    %r15d,%ecx
    0.00 :          19e9b:       mov    $0x39,%r15d
    0.00 :          19ea1:       shl    %cl,%r12
    0.00 :          19ea4:       jmpq   19dc0 <jpeg_fill_bit_buffer+0x40>
    0.00 :          19ea9:       nopl   0x0(%rax)
    0.00 :          19eb0:       add    $0x18,%rsp
    0.00 :          19eb4:       xor    %eax,%eax
    0.00 :          19eb6:       pop    %rbx
    0.00 :          19eb7:       pop    %rbp
    0.00 :          19eb8:       pop    %r12
    0.00 :          19eba:       pop    %r13
    0.00 :          19ebc:       pop    %r14
    0.00 :          19ebe:       pop    %r15
    0.00 :          19ec0:       retq
    0.00 :          19ec1:       mov    (%r14),%rax
    0.00 :          19ec4:       mov    $0xffffffff,%esi
    0.00 :          19ec9:       mov    %r14,%rdi
    0.00 :          19ecc:       movl   $0x78,0x28(%rax)
    0.00 :          19ed3:       callq  *0x8(%rax)
    0.00 :          19ed6:       mov    0x268(%r14),%rax
    0.00 :          19edd:       movl   $0x1,0x10(%rax)
    0.00 :          19ee4:       jmp    19e93 <jpeg_fill_bit_buffer+0x113>
 Percent |       Source code & Disassembly of [jbd2]
------------------------------------------------
 Percent |       Source code & Disassembly of [jbd2]
------------------------------------------------
 Percent |       Source code & Disassembly of libjpeg.so.8.0.2
------------------------------------------------
         :
         :
         :
         :       Disassembly of section .text:
         :
         :       0000000000019ef0 <jpeg_huff_decode>:
    0.00 :          19ef0:       push   %r13
    0.00 :          19ef2:       mov    %rdi,%r13
    0.00 :          19ef5:       push   %r12
    0.00 :          19ef7:       mov    %rcx,%r12
    0.00 :          19efa:       push   %rbp
    0.00 :          19efb:       push   %rbx
    0.00 :          19efc:       sub    $0x18,%rsp
    0.00 :          19f00:       cmp    %edx,%r8d
    0.00 :          19f03:       jg     19fb8 <jpeg_huff_decode+0xc8>
    0.00 :          19f09:       sub    %r8d,%edx
    0.00 :          19f0c:       mov    %rsi,%rdi
    0.00 :          19f0f:       mov    $0x1,%eax
    0.00 :          19f14:       mov    %edx,%ecx
    0.00 :          19f16:       mov    %r8d,%ebx
    0.00 :          19f19:       shr    %cl,%rdi
    0.00 :          19f1c:       mov    %r8d,%ecx
    0.00 :          19f1f:       shl    %cl,%eax
    0.00 :          19f21:       movslq %r8d,%rcx
    0.00 :          19f24:       sub    $0x1,%eax
```

```
 0.00 :        19f27:       and    %edi,%eax
 0.00 :        19f29:       cltq
 0.00 :        19f2b:       cmp    (%r12,%rcx,8),%rax
 0.00 :        19f2f:       jg     19f55 <jpeg_huff_decode+0x65>
 0.00 :        19f31:       jmp    19f80 <jpeg_huff_decode+0x90>
 0.00 :        19f33:       nopl   0x0(%rax,%rax,1)
 0.00 :        19f38:       sub    $0x1,%edx
 0.00 :        19f3b:       mov    %rsi,%rax
 0.00 :        19f3e:       add    $0x1,%ebx
 0.00 :        19f41:       mov    %edx,%ecx
 0.00 :        19f43:       movslq %ebx,%r8
50.00 :        19f46:       shr    %cl,%rax
 0.00 :        19f49:       and    $0x1,%eax
 0.00 :        19f4c:       or     %rbp,%rax
 0.00 :        19f4f:       cmp    %rax,(%r12,%r8,8)
 0.00 :        19f53:       jge    19f80 <jpeg_huff_decode+0x90>
 0.00 :        19f55:       test   %edx,%edx
 0.00 :        19f57:       lea    (%rax,%rax,1),%rbp
 0.00 :        19f5b:       jg     19f38 <jpeg_huff_decode+0x48>
 0.00 :        19f5d:       mov    $0x1,%ecx
 0.00 :        19f62:       mov    %r13,%rdi
 0.00 :        19f65:       callq  4090 <jpeg_fill_bit_buffer@plt>
 0.00 :        19f6a:       test   %eax,%eax
 0.00 :        19f6c:       je     19ffe <jpeg_huff_decode+0x10e>
 0.00 :        19f72:       mov    0x10(%r13),%rsi
 0.00 :        19f76:       mov    0x18(%r13),%edx
 0.00 :        19f7a:       jmp    19f38 <jpeg_huff_decode+0x48>
 0.00 :        19f7c:       nopl   0x0(%rax)
50.00 :        19f80:       cmp    $0x10,%ebx
 0.00 :        19f83:       mov    %rsi,0x10(%r13)
 0.00 :        19f87:       mov    %edx,0x18(%r13)
 0.00 :        19f8b:       jg     19fdb <jpeg_huff_decode+0xeb>
 0.00 :        19f8d:       movslq %ebx,%rbx
 0.00 :        19f90:       mov    0x120(%r12),%rdx
 0.00 :        19f98:       add    0x90(%r12,%rbx,8),%eax
 0.00 :        19fa0:       cltq
 0.00 :        19fa2:       movzbl 0x11(%rdx,%rax,1),%eax
 0.00 :        19fa7:       add    $0x18,%rsp
 0.00 :        19fab:       pop    %rbx
 0.00 :        19fac:       pop    %rbp
 0.00 :        19fad:       pop    %r12
 0.00 :        19faf:       pop    %r13
 0.00 :        19fb1:       retq
 0.00 :        19fb2:       nopw   0x0(%rax,%rax,1)
 0.00 :        19fb8:       mov    %r8d,%ecx
 0.00 :        19fbb:       mov    %r8d,0x8(%rsp)
 0.00 :        19fc0:       callq  4090 <jpeg_fill_bit_buffer@plt>
 0.00 :        19fc5:       test   %eax,%eax
 0.00 :        19fc7:       mov    0x8(%rsp),%r8d
 0.00 :        19fcc:       je     19ffe <jpeg_huff_decode+0x10e>
 0.00 :        19fce:       mov    0x10(%r13),%rsi
 0.00 :        19fd2:       mov    0x18(%r13),%edx
 0.00 :        19fd6:       jmpq   19f09 <jpeg_huff_decode+0x19>
 0.00 :        19fdb:       mov    0x20(%r13),%rdi
 0.00 :        19fdf:       mov    $0xffffffff,%esi
 0.00 :        19fe4:       mov    (%rdi),%rax
 0.00 :        19fe7:       movl   $0x79,0x28(%rax)
 0.00 :        19fee:       callq  *0x8(%rax)
 0.00 :        19ff1:       add    $0x18,%rsp
 0.00 :        19ff5:       xor    %eax,%eax
 0.00 :        19ff7:       pop    %rbx
 0.00 :        19ff8:       pop    %rbp
 0.00 :        19ff9:       pop    %r12
 0.00 :        19ffb:       pop    %r13
```

```
    0.00 :          19ffd:       retq
    0.00 :          19ffe:       add    $0x18,%rsp
    0.00 :          1a002:       mov    $0xffffffff,%eax
    0.00 :          1a007:       pop    %rbx
    0.00 :          1a008:       pop    %rbp
    0.00 :          1a009:       pop    %r12
    0.00 :          1a00b:       pop    %r13
    0.00 :          1a00d:       retq
 Percent |       Source code & Disassembly of libc-2.17.so
-----------------------------------------------
         :
         :
         :
         :       Disassembly of section .text:
         :
         :       000000000007b190 <_int_free>:
    0.00 :          7b190:       push   %r15
    0.00 :          7b192:       push   %r14
    0.00 :          7b194:       push   %r13
    0.00 :          7b196:       push   %r12
    0.00 :          7b198:       push   %rbp
    0.00 :          7b199:       push   %rbx
    0.00 :          7b19a:       mov    %rsi,%rbx
    0.00 :          7b19d:       sub    $0x58,%rsp
    0.00 :          7b1a1:       mov    0x8(%rsi),%rax
    0.00 :          7b1a5:       mov    %edx,0x20(%rsp)
    0.00 :          7b1a9:       mov    %rax,%rbp
    0.00 :          7b1ac:       and    $0xfffffffffffffff8,%rbp
    0.00 :          7b1b0:       mov    %rbp,%rdx
    0.00 :          7b1b3:       neg    %rdx
   50.00 :          7b1b6:       cmp    %rdx,%rsi
    0.00 :          7b1b9:       ja     7b8a7 <_int_free+0x717>
    0.00 :          7b1bf:       test   $0xf,%sil
    0.00 :          7b1c3:       jne    7b8a7 <_int_free+0x717>
    0.00 :          7b1c9:       cmp    $0x1f,%rbp
    0.00 :          7b1cd:       jbe    7b8b0 <_int_free+0x720>
    0.00 :          7b1d3:       test   $0x8,%al
    0.00 :          7b1d5:       jne    7b8b0 <_int_free+0x720>
    0.00 :          7b1db:       cmp    0x32e67e(%rip),%rbp        # 3a9860 <global_max_fast>
    0.00 :          7b1e2:       mov    %rdi,%r15
    0.00 :          7b1e5:       ja     7b2a8 <_int_free+0x118>
    0.00 :          7b1eb:       lea    (%rsi,%rbp,1),%rdx
    0.00 :          7b1ef:       mov    0x8(%rdx),%rax
    0.00 :          7b1f3:       cmp    $0x10,%rax
    0.00 :          7b1f7:       jbe    7b902 <_int_free+0x772>
    0.00 :          7b1fd:       and    $0xfffffffffffffff8,%rax
    0.00 :          7b201:       cmp    0x878(%rdi),%rax
    0.00 :          7b208:       jae    7b902 <_int_free+0x772>
    0.00 :          7b20e:       mov    0x32e654(%rip),%eax        # 3a9868 <perturb_byte>
    0.00 :          7b214:       test   %eax,%eax
    0.00 :          7b216:       jne    7b8b9 <_int_free+0x729>
    0.00 :          7b21c:       cmpl   $0x0,%fs:0x18
    0.00 :          7b225:       je     7b228 <_int_free+0x98>
    0.00 :          7b227:       lock andl $0xfffffffe,0x4(%r15)
    0.00 :          7b22d:       shr    $0x4,%ebp
    0.00 :          7b230:       sub    $0x2,%ebp
    0.00 :          7b233:       mov    %ebp,%eax
    0.00 :          7b235:       mov    0x8(%r15,%rax,8),%rcx
    0.00 :          7b23a:       lea    0x8(%r15,%rax,8),%rdx
    0.00 :          7b23f:       cmp    %rcx,%rbx
    0.00 :          7b242:       je     7b86d <_int_free+0x6dd>
    0.00 :          7b248:       mov    $0xffffffff,%esi
    0.00 :          7b24d:       jmp    7b25c <_int_free+0xcc>
    0.00 :          7b24f:       nop
```

```
 0.00 :        7b250:       cmp    %rax,%rbx
 0.00 :        7b253:       mov    %rax,%rcx
 0.00 :        7b256:       je     7b86d <_int_free+0x6dd>
 0.00 :        7b25c:       test   %rcx,%rcx
 0.00 :        7b25f:       je     7b26a <_int_free+0xda>
 0.00 :        7b261:       mov    0x8(%rcx),%esi
 0.00 :        7b264:       shr    $0x4,%esi
 0.00 :        7b267:       sub    $0x2,%esi
 0.00 :        7b26a:       mov    %rcx,0x10(%rbx)
 0.00 :        7b26e:       mov    %rcx,%rax
 0.00 :        7b271:       cmpl   $0x0,%fs:0x18
 0.00 :        7b27a:       je     7b27d <_int_free+0xed>
 0.00 :        7b27c:       lock cmpxchg %rbx,(%rdx)
 0.00 :        7b281:       cmp    %rax,%rcx
 0.00 :        7b284:       jne    7b250 <_int_free+0xc0>
 0.00 :        7b286:       test   %rcx,%rcx
 0.00 :        7b289:       je     7b293 <_int_free+0x103>
 0.00 :        7b28b:       cmp    %ebp,%esi
 0.00 :        7b28d:       jne    7b9c6 <_int_free+0x836>
 0.00 :        7b293:       add    $0x58,%rsp
 0.00 :        7b297:       pop    %rbx
 0.00 :        7b298:       pop    %rbp
 0.00 :        7b299:       pop    %r12
 0.00 :        7b29b:       pop    %r13
 0.00 :        7b29d:       pop    %r14
 0.00 :        7b29f:       pop    %r15
 0.00 :        7b2a1:       retq
 0.00 :        7b2a2:       nopw   0x0(%rax,%rax,1)
 0.00 :        7b2a8:       test   $0x2,%al
 0.00 :        7b2aa:       jne    7b730 <_int_free+0x5a0>
 0.00 :        7b2b0:       mov    0x20(%rsp),%r9d
 0.00 :        7b2b5:       movl   $0x0,0x24(%rsp)
 0.00 :        7b2bd:       test   %r9d,%r9d
 0.00 :        7b2c0:       je     7b600 <_int_free+0x470>
 0.00 :        7b2c6:       mov    0x58(%r15),%rax
 0.00 :        7b2ca:       lea    (%rbx,%rbp,1),%r12
 0.00 :        7b2ce:       cmp    %rax,%rbx
 0.00 :        7b2d1:       je     7bc3f <_int_free+0xaaf>
 0.00 :        7b2d7:       testb  $0x2,0x4(%r15)
 0.00 :        7b2dc:       je     7bc4b <_int_free+0xabb>
 0.00 :        7b2e2:       mov    0x8(%r12),%rax
 0.00 :        7b2e7:       test   $0x1,%al
 0.00 :        7b2e9:       je     7bc6b <_int_free+0xadb>
 0.00 :        7b2ef:       mov    %rax,%r13
 0.00 :        7b2f2:       and    $0xfffffffffffffff8,%r13
 0.00 :        7b2f6:       cmp    $0x10,%rax
 0.00 :        7b2fa:       jbe    7b9a2 <_int_free+0x812>
 0.00 :        7b300:       cmp    0x878(%r15),%r13
 0.00 :        7b307:       jae    7b9a2 <_int_free+0x812>
 0.00 :        7b30d:       mov    0x32e555(%rip),%eax        # 3a9868 <perturb_byte>
 0.00 :        7b313:       test   %eax,%eax
 0.00 :        7b315:       jne    7bc77 <_int_free+0xae7>
 0.00 :        7b31b:       testb  $0x1,0x8(%rbx)
 0.00 :        7b31f:       jne    7b365 <_int_free+0x1d5>
 0.00 :        7b321:       mov    (%rbx),%rax
 0.00 :        7b324:       sub    %rax,%rbx
 0.00 :        7b327:       add    %rax,%rbp
 0.00 :        7b32a:       mov    0x10(%rbx),%rax
 0.00 :        7b32e:       mov    0x18(%rbx),%rdx
 0.00 :        7b332:       cmp    0x18(%rax),%rbx
 0.00 :        7b336:       jne    7bc25 <_int_free+0xa95>
 0.00 :        7b33c:       cmp    0x10(%rdx),%rbx
 0.00 :        7b340:       jne    7bc25 <_int_free+0xa95>
 0.00 :        7b346:       cmpq   $0x3ff,0x8(%rbx)
```

```
  0.00 :        7b34e:       mov    %rdx,0x18(%rax)
  0.00 :        7b352:       mov    %rax,0x10(%rdx)
  0.00 :        7b356:       jbe    7b365 <_int_free+0x1d5>
  0.00 :        7b358:       mov    0x20(%rbx),%rdx
  0.00 :        7b35c:       test   %rdx,%rdx
  0.00 :        7b35f:       jne    7bb1f <_int_free+0x98f>
  0.00 :        7b365:       cmp    %r12,0x58(%r15)
  0.00 :        7b369:       je     7b760 <_int_free+0x5d0>
  0.00 :        7b36f:       testb  $0x1,0x8(%r12,%r13,1)
  0.00 :        7b375:       jne    7b640 <_int_free+0x4b0>
  0.00 :        7b37b:       mov    0x10(%r12),%rax
  0.00 :        7b380:       mov    0x18(%r12),%rdx
  0.00 :        7b385:       cmp    0x18(%rax),%r12
  0.00 :        7b389:       jne    7bb05 <_int_free+0x975>
  0.00 :        7b38f:       cmp    0x10(%rdx),%r12
  0.00 :        7b393:       jne    7bb05 <_int_free+0x975>
  0.00 :        7b399:       cmpq   $0x3ff,0x8(%r12)
  0.00 :        7b3a2:       mov    %rdx,0x18(%rax)
  0.00 :        7b3a6:       mov    %rax,0x10(%rdx)
  0.00 :        7b3aa:       jbe    7b3ba <_int_free+0x22a>
  0.00 :        7b3ac:       mov    0x20(%r12),%rdx
  0.00 :        7b3b1:       test   %rdx,%rdx
  0.00 :        7b3b4:       jne    7bad0 <_int_free+0x940>
  0.00 :        7b3ba:       add    %r13,%rbp
  0.00 :        7b3bd:       mov    0x68(%r15),%rax
  0.00 :        7b3c1:       lea    0x58(%r15),%rdx
  0.00 :        7b3c5:       cmp    0x18(%rax),%rdx
  0.00 :        7b3c9:       jne    7bac4 <_int_free+0x934>
  0.00 :        7b3cf:       cmp    $0x3ff,%rbp
  0.00 :        7b3d6:       mov    %rax,0x10(%rbx)
  0.00 :        7b3da:       mov    %rdx,0x18(%rbx)
  0.00 :        7b3de:       jbe    7b3f0 <_int_free+0x260>
  0.00 :        7b3e0:       movq   $0x0,0x20(%rbx)
  0.00 :        7b3e8:       movq   $0x0,0x28(%rbx)
  0.00 :        7b3f0:       mov    %rbx,0x18(%rax)
  0.00 :        7b3f4:       mov    %rbp,%rax
  0.00 :        7b3f7:       mov    %rbx,0x68(%r15)
  0.00 :        7b3fb:       or     $0x1,%rax
  0.00 :        7b3ff:       mov    %rbp,(%rbx,%rbp,1)
  0.00 :        7b403:       mov    %rax,0x8(%rbx)
  0.00 :        7b407:       cmp    $0xffff,%rbp
  0.00 :        7b40e:       jbe    7b6e8 <_int_free+0x558>
  0.00 :        7b414:       testb  $0x1,0x4(%r15)
  0.00 :        7b419:       je     7b780 <_int_free+0x5f0>
  0.00 :        7b41f:       lea    0x32c21a(%rip),%rax        # 3a7640 <main_arena>
  0.00 :        7b426:       cmp    %rax,%r15
  0.00 :        7b429:       je     7b78d <_int_free+0x5fd>
  0.00 :        7b42f:       mov    0x58(%r15),%rax
  0.00 :        7b433:       mov    %rax,%rdi
  0.00 :        7b436:       and    $0xfffffffffc000000,%rdi
  0.00 :        7b43d:       mov    (%rdi),%r12
  0.00 :        7b440:       cmp    %r15,%r12
  0.00 :        7b443:       jne    7b9f1 <_int_free+0x861>
  0.00 :        7b449:       mov    0x32b9d8(%rip),%rdx        # 3a6e28 <_DYNAMIC+0x2e8>
  0.00 :        7b450:       mov    0x32bd11(%rip),%rcx        # 3a7168 <mp_+0x8>
  0.00 :        7b457:       lea    0x20(%rdi),%rbx
  0.00 :        7b45b:       cmp    %rax,%rbx
  0.00 :        7b45e:       mov    0x18(%rdx),%rdx
  0.00 :        7b462:       mov    %rcx,0x30(%rsp)
  0.00 :        7b467:       mov    %rdx,0x28(%rsp)
  0.00 :        7b46c:       jne    7b7c8 <_int_free+0x638>
  0.00 :        7b472:       mov    0x8(%rdi),%rbp
  0.00 :        7b476:       mov    0x10(%rbp),%rsi
  0.00 :        7b47a:       lea    -0x10(%rsi),%rdx
```

```
 0.00 :        7b47e:       lea    0x0(%rbp,%rdx,1),%rax
 0.00 :        7b483:       and    $0xf,%eax
 0.00 :        7b486:       sub    %rax,%rdx
 0.00 :        7b489:       add    %rbp,%rdx
 0.00 :        7b48c:       cmpq   $0x1,0x8(%rdx)
 0.00 :        7b491:       jne    7b7f6 <_int_free+0x666>
 0.00 :        7b497:       mov    0x28(%rsp),%r8
 0.00 :        7b49c:       mov    0x30(%rsp),%rcx
 0.00 :        7b4a1:       mov    %r15,0x38(%rsp)
 0.00 :        7b4a6:       mov    %r12,%r15
 0.00 :        7b4a9:       lea    0x20(%rcx,%r8,1),%r14
 0.00 :        7b4ae:       sub    $0x1,%r8
 0.00 :        7b4b2:       mov    %r8,0x18(%rsp)
 0.00 :        7b4b7:       jmpq   7b576 <_int_free+0x3e6>
 0.00 :        7b4bc:       nopl   0x0(%rax)
 0.00 :        7b4c0:       mov    $0x4000000,%esi
 0.00 :        7b4c5:       mov    %r12,%rbx
 0.00 :        7b4c8:       callq  e4d00 <munmap>
 0.00 :        7b4cd:       testb  $0x1,0x8(%r12)
 0.00 :        7b4d3:       jne    7b514 <_int_free+0x384>
 0.00 :        7b4d5:       sub    (%r12),%rbx
 0.00 :        7b4d9:       mov    0x10(%rbx),%rax
 0.00 :        7b4dd:       mov    0x18(%rbx),%rdx
 0.00 :        7b4e1:       cmp    0x18(%rax),%rbx
 0.00 :        7b4e5:       jne    7b834 <_int_free+0x6a4>
 0.00 :        7b4eb:       cmp    0x10(%rdx),%rbx
 0.00 :        7b4ef:       jne    7b834 <_int_free+0x6a4>
 0.00 :        7b4f5:       cmpq   $0x3ff,0x8(%rbx)
 0.00 :        7b4fd:       mov    %rdx,0x18(%rax)
 0.00 :        7b501:       mov    %rax,0x10(%rdx)
 0.00 :        7b505:       jbe    7b514 <_int_free+0x384>
 0.00 :        7b507:       mov    0x20(%rbx),%rdx
 0.00 :        7b50b:       test   %rdx,%rdx
 0.00 :        7b50e:       jne    7b8ce <_int_free+0x73e>
 0.00 :        7b514:       lea    (%rbx,%r13,1),%rax
 0.00 :        7b518:       test   %rax,0x18(%rsp)
 0.00 :        7b51d:       jne    7b815 <_int_free+0x685>
 0.00 :        7b523:       mov    %rbp,%rdx
 0.00 :        7b526:       add    0x10(%rbp),%rdx
 0.00 :        7b52a:       cmp    %rdx,%rax
 0.00 :        7b52d:       jne    7b888 <_int_free+0x6f8>
 0.00 :        7b533:       lea    0x20(%rbp),%rax
 0.00 :        7b537:       or     $0x1,%r13
 0.00 :        7b53b:       mov    %rbx,0x58(%r15)
 0.00 :        7b53f:       mov    %r13,0x8(%rbx)
 0.00 :        7b543:       cmp    %rax,%rbx
 0.00 :        7b546:       jne    7b750 <_int_free+0x5c0>
 0.00 :        7b54c:       mov    0x8(%rbp),%rcx
 0.00 :        7b550:       mov    %rbp,%rdi
 0.00 :        7b553:       mov    0x10(%rcx),%rsi
 0.00 :        7b557:       lea    -0x10(%rsi),%rdx
 0.00 :        7b55b:       lea    (%rcx,%rdx,1),%rax
 0.00 :        7b55f:       and    $0xf,%eax
 0.00 :        7b562:       sub    %rax,%rdx
 0.00 :        7b565:       add    %rcx,%rdx
 0.00 :        7b568:       cmpq   $0x1,0x8(%rdx)
 0.00 :        7b56d:       jne    7b7f6 <_int_free+0x666>
 0.00 :        7b573:       mov    %rcx,%rbp
 0.00 :        7b576:       mov    %rdx,%r12
 0.00 :        7b579:       sub    (%rdx),%r12
 0.00 :        7b57c:       mov    0x8(%r12),%rcx
 0.00 :        7b581:       mov    %rcx,%r9
 0.00 :        7b584:       and    $0xfffffffffffffff8,%r9
 0.00 :        7b588:       add    %r9,%rax
```

```
 0.00 :        7b58b:       lea    0x10(%rax),%r13
 0.00 :        7b58f:       add    $0xf,%rax
 0.00 :        7b593:       cmp    $0x3e,%rax
 0.00 :        7b597:       ja     7b7d7 <_int_free+0x647>
 0.00 :        7b59d:       and    $0x1,%ecx
 0.00 :        7b5a0:       jne    7b5a6 <_int_free+0x416>
 0.00 :        7b5a2:       add    (%r12),%r13
 0.00 :        7b5a6:       lea    -0x1(%r13),%rax
 0.00 :        7b5aa:       cmp    $0x3fffffe,%rax
 0.00 :        7b5b0:       ja     7b84e <_int_free+0x6be>
 0.00 :        7b5b6:       mov    $0x4000000,%eax
 0.00 :        7b5bb:       sub    %rsi,%rax
 0.00 :        7b5be:       add    %r13,%rax
 0.00 :        7b5c1:       cmp    %r14,%rax
 0.00 :        7b5c4:       jb     7b650 <_int_free+0x4c0>
 0.00 :        7b5ca:       mov    0x10(%rdi),%rax
 0.00 :        7b5ce:       sub    %rax,0x878(%r15)
 0.00 :        7b5d5:       sub    %rax,0x32e294(%rip)        # 3a9870 <arena_mem>
 0.00 :        7b5dc:       lea    0x4000000(%rdi),%rax
 0.00 :        7b5e3:       cmp    %rax,0x32e28e(%rip)        # 3a9878 <aligned_heap_area>
 0.00 :        7b5ea:       jne    7b4c0 <_int_free+0x330>
 0.00 :        7b5f0:       movq   $0x0,0x32e27d(%rip)        # 3a9878 <aligned_heap_area>
 0.00 :        7b5fb:       jmpq   7b4c0 <_int_free+0x330>
 0.00 :        7b600:       mov    $0x1,%esi
 0.00 :        7b605:       mov    0x20(%rsp),%eax
 0.00 :        7b609:       cmpl   $0x0,0x331444(%rip)        # 3aca54 <__libc_multiple_threads>
 0.00 :        7b610:       je     7b61f <_int_free+0x48f>
 0.00 :        7b612:       lock cmpxchg %esi,(%r15)
 0.00 :        7b617:       jne    8063b <_L_lock_2370>
 0.00 :        7b61d:       jmp    7b629 <_int_free+0x499>
 0.00 :        7b61f:       cmpxchg %esi,(%r15)
 0.00 :        7b623:       jne    8063b <_L_lock_2370>
 0.00 :        7b629:       movl   $0x1,0x24(%rsp)
 0.00 :        7b631:       jmpq   7b2c6 <_int_free+0x136>
 0.00 :        7b636:       nopw   %cs:0x0(%rax,%rax,1)
 0.00 :        7b640:       andq   $0xfffffffffffffffe,0x8(%r12)
 0.00 :        7b646:       jmpq   7b3bd <_int_free+0x22d>
 0.00 :        7b64b:       nopl   0x0(%rax,%rax,1)
 0.00 :        7b650:       mov    %r15,%r12
 0.00 :        7b653:       mov    0x8(%rbx),%rcx
 0.00 :        7b657:       mov    0x38(%rsp),%r15
 0.00 :        7b65c:       mov    %rdi,%rbp
 0.00 :        7b65f:       mov    %rcx,%r14
 0.00 :        7b662:       and    $0xfffffffffffffff8,%r14
 0.00 :        7b666:       mov    %r14,%rax
 0.00 :        7b669:       sub    0x30(%rsp),%rax
 0.00 :        7b66e:       lea    -0x21(%rax),%r13
 0.00 :        7b672:       mov    0x28(%rsp),%rax
 0.00 :        7b677:       neg    %rax
 0.00 :        7b67a:       and    %rax,%r13
 0.00 :        7b67d:       cmp    0x28(%rsp),%r13
 0.00 :        7b682:       jl     7b6e8 <_int_free+0x558>
 0.00 :        7b684:       mov    0x10(%rbp),%r10
 0.00 :        7b688:       sub    %r13,%r10
 0.00 :        7b68b:       cmp    $0x1f,%r10
 0.00 :        7b68f:       jle    7b6e8 <_int_free+0x558>
 0.00 :        7b691:       cmpl   $0x0,0x32bb24(%rip)        # 3a71bc <may_shrink_heap.10739>
 0.00 :        7b698:       jl     7ba43 <_int_free+0x8b3>
 0.00 :        7b69e:       setne  %al
 0.00 :        7b6a1:       test   %al,%al
 0.00 :        7b6a3:       lea    0x0(%rbp,%r10,1),%rdi
 0.00 :        7b6a8:       jne    7ba10 <_int_free+0x880>
 0.00 :        7b6ae:       mov    $0x4,%edx
 0.00 :        7b6b3:       mov    %r13,%rsi
```

```
 0.00 :        7b6b6:       mov    %r10,0x8(%rsp)
 0.00 :        7b6bb:       callq  e4dc0 <__madvise>
 0.00 :        7b6c0:       mov    0x8(%rsp),%r10
 0.00 :        7b6c5:       sub    %r13,0x32e1a4(%rip)        # 3a9870 <arena_mem>
 0.00 :        7b6cc:       sub    %r13,%r14
 0.00 :        7b6cf:       mov    %r10,0x10(%rbp)
 0.00 :        7b6d3:       or     $0x1,%r14
 0.00 :        7b6d7:       sub    %r13,0x878(%r12)
 0.00 :        7b6df:       mov    %r14,0x8(%rbx)
 0.00 :        7b6e3:       nopl   0x0(%rax,%rax,1)
 0.00 :        7b6e8:       mov    0x20(%rsp),%r8d
 0.00 :        7b6ed:       test   %r8d,%r8d
 0.00 :        7b6f0:       jne    7b293 <_int_free+0x103>
 0.00 :        7b6f6:       mov    0x24(%rsp),%edi
 0.00 :        7b6fa:       test   %edi,%edi
 0.00 :        7b6fc:       je     7b9d2 <_int_free+0x842>
 0.00 :        7b702:       cmpl   $0x0,0x33134b(%rip)        # 3aca54 <__libc_multiple_threads>
 0.00 :        7b709:       je     7b717 <_int_free+0x587>
 0.00 :        7b70b:       lock decl (%r15)
 0.00 :        7b70f:       jne    80656 <_L_unlock_2438>
 0.00 :        7b715:       jmp    7b720 <_int_free+0x590>
 0.00 :        7b717:       decl   (%r15)
 0.00 :        7b71a:       jne    80656 <_L_unlock_2438>
 0.00 :        7b720:       add    $0x58,%rsp
 0.00 :        7b724:       pop    %rbx
 0.00 :        7b725:       pop    %rbp
 0.00 :        7b726:       pop    %r12
 0.00 :        7b728:       pop    %r13
 0.00 :        7b72a:       pop    %r14
 0.00 :        7b72c:       pop    %r15
 0.00 :        7b72e:       retq
 0.00 :        7b72f:       nop
 0.00 :        7b730:       mov    %rsi,%rdi
 0.00 :        7b733:       callq  7b110 <munmap_chunk>
 0.00 :        7b738:       add    $0x58,%rsp
 0.00 :        7b73c:       pop    %rbx
 0.00 :        7b73d:       pop    %rbp
 0.00 :        7b73e:       pop    %r12
 0.00 :        7b740:       pop    %r13
 0.00 :        7b742:       pop    %r14
 0.00 :        7b744:       pop    %r15
 0.00 :        7b746:       retq
 0.00 :        7b747:       nopw   0x0(%rax,%rax,1)
 0.00 :        7b750:       mov    %r15,%r12
 0.00 :        7b753:       mov    %r13,%rcx
 0.00 :        7b756:       mov    0x38(%rsp),%r15
 0.00 :        7b75b:       jmpq   7b65f <_int_free+0x4cf>
 0.00 :        7b760:       add    %r13,%rbp
 0.00 :        7b763:       mov    %rbp,%rax
 0.00 :        7b766:       or     $0x1,%rax
 0.00 :        7b76a:       mov    %rax,0x8(%rbx)
 0.00 :        7b76e:       mov    %rbx,0x58(%r15)
 0.00 :        7b772:       jmpq   7b407 <_int_free+0x277>
 0.00 :        7b777:       nopw   0x0(%rax,%rax,1)
 0.00 :        7b780:       mov    %r15,%rdi
 0.00 :        7b783:       callq  7ab30 <malloc_consolidate>
 0.00 :        7b788:       jmpq   7b41f <_int_free+0x28f>
 0.00 :        7b78d:       mov    0x32bf04(%rip),%rax        # 3a7698 <main_arena+0x58>
 0.00 :        7b794:       mov    0x8(%rax),%rax
 0.00 :        7b798:       and    $0xfffffffffffffff8,%rax
 0.00 :        7b79c:       cmp    0x32b9bd(%rip),%rax        # 3a7160 <mp_>
 0.00 :        7b7a3:       jb     7b6e8 <_int_free+0x558>
 0.00 :        7b7a9:       mov    0x32b9b8(%rip),%rdi        # 3a7168 <mp_+0x8>
 0.00 :        7b7b0:       lea    0x32c701(%rip),%rdx        # 3a7eb8 <main_arena+0x878>
```

```
 0.00 :        7b7b7:       lea    0x32beda(%rip),%rsi        # 3a7698 <main_arena+0x58>
 0.00 :        7b7be:       callq  7a960 <systrim.isra.1>
 0.00 :        7b7c3:       jmpq   7b6e8 <_int_free+0x558>
 0.00 :        7b7c8:       mov    0x8(%rax),%rcx
 0.00 :        7b7cc:       mov    %rax,%rbx
 0.00 :        7b7cf:       mov    %rdi,%rbp
 0.00 :        7b7d2:       jmpq   7b65f <_int_free+0x4cf>
 0.00 :        7b7d7:       lea    0xed8a2(%rip),%rcx        # 169080 <__func__.10900>
 0.00 :        7b7de:       lea    0xed645(%rip),%rsi        # 168e2a <__PRETTY_FUNCTION__.11753+0x18d>
 0.00 :        7b7e5:       lea    0xf183c(%rip),%rdi        # 16d028 <__PRETTY_FUNCTION__.9328+0x1c00>
 0.00 :        7b7ec:       mov    $0x2a2,%edx
 0.00 :        7b7f1:       callq  7a110 <__malloc_assert>
 0.00 :        7b7f6:       lea    0xed883(%rip),%rcx        # 169080 <__func__.10900>
 0.00 :        7b7fd:       lea    0xed626(%rip),%rsi        # 168e2a <__PRETTY_FUNCTION__.11753+0x18d>
 0.00 :        7b804:       lea    0xed627(%rip),%rdi        # 168e32 <__PRETTY_FUNCTION__.11753+0x195>
 0.00 :        7b80b:       mov    $0x29f,%edx
 0.00 :        7b810:       callq  7a110 <__malloc_assert>
 0.00 :        7b815:       lea    0xed864(%rip),%rcx        # 169080 <__func__.10900>
 0.00 :        7b81c:       lea    0xed607(%rip),%rsi        # 168e2a <__PRETTY_FUNCTION__.11753+0x18d>
 0.00 :        7b823:       lea    0xf18ee(%rip),%rdi        # 16d118 <__PRETTY_FUNCTION__.9328+0x1cf0>
 0.00 :        7b82a:       mov    $0x2b0,%edx
 0.00 :        7b82f:       callq  7a110 <__malloc_assert>
 0.00 :        7b834:       mov    0x32b97e(%rip),%edi        # 3a71b8 <check_action>
 0.00 :        7b83a:       lea    0xed552(%rip),%rsi        # 168d93 <__PRETTY_FUNCTION__.11753+0xf6>
 0.00 :        7b841:       mov    %rbx,%rdx
 0.00 :        7b844:       callq  7aa50 <malloc_printerr>
 0.00 :        7b849:       jmpq   7b514 <_int_free+0x384>
 0.00 :        7b84e:       lea    0xed82b(%rip),%rcx        # 169080 <__func__.10900>
 0.00 :        7b855:       lea    0xed5ce(%rip),%rsi        # 168e2a <__PRETTY_FUNCTION__.11753+0x18d>
 0.00 :        7b85c:       lea    0xf1875(%rip),%rdi        # 16d0d8 <__PRETTY_FUNCTION__.9328+0x1cb0>
 0.00 :        7b863:       mov    $0x2a5,%edx
 0.00 :        7b868:       callq  7a110 <__malloc_assert>
 0.00 :        7b86d:       lea    0xf178c(%rip),%rsi        # 16d000 <__PRETTY_FUNCTION__.9328+0x1bd8>
 0.00 :        7b874:       mov    0x32b93e(%rip),%edi        # 3a71b8 <check_action>
 0.00 :        7b87a:       lea    0x10(%rbx),%rdx
 0.00 :        7b87e:       callq  7aa50 <malloc_printerr>
 0.00 :        7b883:       jmpq   7b293 <_int_free+0x103>
 0.00 :        7b888:       lea    0xed7f1(%rip),%rcx        # 169080 <__func__.10900>
 0.00 :        7b88f:       lea    0xed594(%rip),%rsi        # 168e2a <__PRETTY_FUNCTION__.11753+0x18d>
 0.00 :        7b896:       lea    0xf18bb(%rip),%rdi        # 16d158 <__PRETTY_FUNCTION__.9328+0x1d30>
 0.00 :        7b89d:       mov    $0x2b1,%edx
 0.00 :        7b8a2:       callq  7a110 <__malloc_assert>
 0.00 :        7b8a7:       lea    0xed51f(%rip),%rsi        # 168dcd <__PRETTY_FUNCTION__.11753+0x130>
 0.00 :        7b8ae:       jmp    7b874 <_int_free+0x6e4>
 0.00 :        7b8b0:       lea    0xed52e(%rip),%rsi        # 168de5 <__PRETTY_FUNCTION__.11753+0x148>
 0.00 :        7b8b7:       jmp    7b874 <_int_free+0x6e4>
 0.00 :        7b8b9:       lea    -0x10(%rbp),%rdx
 0.00 :        7b8bd:       lea    0x10(%rbx),%rdi
 0.00 :        7b8c1:       movzbl %al,%esi
 0.00 :        7b8c4:       callq  86cd0 <__GI_memset>
 0.00 :        7b8c9:       jmpq   7b21c <_int_free+0x8c>
 0.00 :        7b8ce:       cmp    0x28(%rdx),%rbx
 0.00 :        7b8d2:       jne    7bccd <_int_free+0xb3d>
 0.00 :        7b8d8:       mov    0x28(%rbx),%rcx
 0.00 :        7b8dc:       cmp    0x20(%rcx),%rbx
 0.00 :        7b8e0:       jne    7bcae <_int_free+0xb1e>
 0.00 :        7b8e6:       cmpq   $0x0,0x20(%rax)
 0.00 :        7b8eb:       je     7bc8c <_int_free+0xafc>
 0.00 :        7b8f1:       mov    %rcx,0x28(%rdx)
 0.00 :        7b8f5:       mov    0x28(%rbx),%rax
 0.00 :        7b8f9:       mov    %rdx,0x20(%rax)
 0.00 :        7b8fd:       jmpq   7b514 <_int_free+0x384>
 0.00 :        7b902:       cmpl   $0x0,0x20(%rsp)
 0.00 :        7b907:       jne    7b96c <_int_free+0x7dc>
```

```
  0.00 :         7b909:       mov    $0x1,%esi
  0.00 :         7b90e:       mov    0x20(%rsp),%eax
  0.00 :         7b912:       cmpl   $0x0,0x33113b(%rip)        # 3aca54 <__libc_multiple_threads>
  0.00 :         7b919:       je     7b928 <_int_free+0x798>
  0.00 :         7b91b:       lock cmpxchg %esi,(%r15)
  0.00 :         7b920:       jne    80671 <_L_lock_2624>
  0.00 :         7b926:       jmp    7b932 <_int_free+0x7a2>
  0.00 :         7b928:       cmpxchg %esi,(%r15)
  0.00 :         7b92c:       jne    80671 <_L_lock_2624>
  0.00 :         7b932:       mov    0x8(%rdx),%rax
  0.00 :         7b936:       cmp    $0x10,%rax
  0.00 :         7b93a:       jbe    7b978 <_int_free+0x7e8>
  0.00 :         7b93c:       and    $0xfffffffffffffff8,%rax
  0.00 :         7b940:       cmp    0x878(%r15),%rax
  0.00 :         7b947:       jae    7b978 <_int_free+0x7e8>
  0.00 :         7b949:       cmpl   $0x0,0x331104(%rip)        # 3aca54 <__libc_multiple_threads>
  0.00 :         7b950:       je     7b95e <_int_free+0x7ce>
  0.00 :         7b952:       lock decl (%r15)
  0.00 :         7b956:       jne    8068c <_L_unlock_2633>
  0.00 :         7b95c:       jmp    7b967 <_int_free+0x7d7>
  0.00 :         7b95e:       decl   (%r15)
  0.00 :         7b961:       jne    8068c <_L_unlock_2633>
  0.00 :         7b967:       jmpq   7b20e <_int_free+0x7e>
  0.00 :         7b96c:       lea    0xf1665(%rip),%rsi        # 16cfd8 <__PRETTY_FUNCTION__.9328+0x1bb0>
  0.00 :         7b973:       jmpq   7b874 <_int_free+0x6e4>
  0.00 :         7b978:       lea    0xf1659(%rip),%rsi        # 16cfd8 <__PRETTY_FUNCTION__.9328+0x1bb0>
  0.00 :         7b97f:       cmpl   $0x0,0x3310ce(%rip)        # 3aca54 <__libc_multiple_threads>
  0.00 :         7b986:       je     7b994 <_int_free+0x804>
  0.00 :         7b988:       lock decl (%r15)
  0.00 :         7b98c:       jne    806a7 <_L_unlock_2646>
  0.00 :         7b992:       jmp    7b99d <_int_free+0x80d>
  0.00 :         7b994:       decl   (%r15)
  0.00 :         7b997:       jne    806a7 <_L_unlock_2646>
  0.00 :         7b99d:       jmpq   7b874 <_int_free+0x6e4>
  0.00 :         7b9a2:       lea    0xf1607(%rip),%rsi        # 16cfb0 <__PRETTY_FUNCTION__.9328+0x1b88>
  0.00 :         7b9a9:       mov    0x24(%rsp),%r11d
  0.00 :         7b9ae:       test   %r11d,%r11d
  0.00 :         7b9b1:       je     7b874 <_int_free+0x6e4>
  0.00 :         7b9b7:       mov    0x20(%rsp),%r10d
  0.00 :         7b9bc:       test   %r10d,%r10d
  0.00 :         7b9bf:       je     7b97f <_int_free+0x7ef>
  0.00 :         7b9c1:       jmpq   7b874 <_int_free+0x6e4>
  0.00 :         7b9c6:       lea    0xed42d(%rip),%rsi        # 168dfa <__PRETTY_FUNCTION__.11753+0x15d>
  0.00 :         7b9cd:       jmpq   7b874 <_int_free+0x6e4>
  0.00 :         7b9d2:       lea    0xed69d(%rip),%rcx        # 169076 <__func__.11466>
  0.00 :         7b9d9:       lea    0xed375(%rip),%rsi        # 168d55 <__PRETTY_FUNCTION__.11753+0xb8>
  0.00 :         7b9e0:       lea    0xed45e(%rip),%rdi        # 168e45 <__PRETTY_FUNCTION__.11753+0x1a8>
  0.00 :         7b9e7:       mov    $0xf98,%edx
  0.00 :         7b9ec:       callq  7a110 <__malloc_assert>
  0.00 :         7b9f1:       lea    0xed67e(%rip),%rcx        # 169076 <__func__.11466>
  0.00 :         7b9f8:       lea    0xed356(%rip),%rsi        # 168d55 <__PRETTY_FUNCTION__.11753+0xb8>
  0.00 :         7b9ff:       lea    0xed411(%rip),%rdi        # 168e17 <__PRETTY_FUNCTION__.11753+0x17a>
  0.00 :         7ba06:       mov    $0xf92,%edx
  0.00 :         7ba0b:       callq  7a110 <__malloc_assert>
  0.00 :         7ba10:       xor    %r9d,%r9d
  0.00 :         7ba13:       or     $0xffffffff,%r8d
  0.00 :         7ba17:       xor    %edx,%edx
  0.00 :         7ba19:       mov    $0x32,%ecx
  0.00 :         7ba1e:       mov    %r13,%rsi
  0.00 :         7ba21:       mov    %r10,0x8(%rsp)
  0.00 :         7ba26:       callq  e4cd0 <mmap>
  0.00 :         7ba2b:       add    $0x1,%rax
  0.00 :         7ba2f:       mov    0x8(%rsp),%r10
  0.00 :         7ba34:       je     7b6e8 <_int_free+0x558>
```

```
 0.00 :        7ba3a:       mov    %r10,0x18(%rbp)
 0.00 :        7ba3e:       jmpq   7b6c5 <_int_free+0x535>
 0.00 :        7ba43:       mov    0x32b39e(%rip),%rax        # 3a6de8 <_DYNAMIC+0x2a8>
 0.00 :        7ba4a:       mov    (%rax),%eax
 0.00 :        7ba4c:       test   %eax,%eax
 0.00 :        7ba4e:       mov    %eax,0x32b768(%rip)        # 3a71bc <may_shrink_heap.10739>
 0.00 :        7ba54:       jne    7bab1 <_int_free+0x921>
 0.00 :        7ba56:       lea    0xf1733(%rip),%rdi        # 16d190 <__PRETTY_FUNCTION__.9328+0x1d68>
 0.00 :        7ba5d:       mov    $0x80000,%esi
 0.00 :        7ba62:       mov    %r10,0x8(%rsp)
 0.00 :        7ba67:       callq  dbea9 <__open_nocancel>
 0.00 :        7ba6c:       test   %eax,%eax
 0.00 :        7ba6e:       mov    0x8(%rsp),%r10
 0.00 :        7ba73:       js     7bab1 <_int_free+0x921>
 0.00 :        7ba75:       lea    0x4f(%rsp),%rsi
 0.00 :        7ba7a:       mov    $0x1,%edx
 0.00 :        7ba7f:       mov    %eax,%edi
 0.00 :        7ba81:       mov    %eax,0x10(%rsp)
 0.00 :        7ba85:       callq  dc099 <__read_nocancel>
 0.00 :        7ba8a:       test   %eax,%eax
 0.00 :        7ba8c:       mov    0x10(%rsp),%ecx
 0.00 :        7ba90:       mov    0x8(%rsp),%r10
 0.00 :        7ba95:       jle    7bac0 <_int_free+0x930>
 0.00 :        7ba97:       xor    %eax,%eax
 0.00 :        7ba99:       cmpb   $0x32,0x4f(%rsp)
 0.00 :        7ba9e:       sete   %al
 0.00 :        7baa1:       mov    %eax,0x32b715(%rip)        # 3a71bc <may_shrink_heap.10739>
 0.00 :        7baa7:       movslq %ecx,%rdi
 0.00 :        7baaa:       mov    $0x3,%eax
 0.00 :        7baaf:       syscall
 0.00 :        7bab1:       cmpl   $0x0,0x32b704(%rip)        # 3a71bc <may_shrink_heap.10739>
 0.00 :        7bab8:       setne  %al
 0.00 :        7babb:       jmpq   7b6a1 <_int_free+0x511>
 0.00 :        7bac0:       xor    %eax,%eax
 0.00 :        7bac2:       jmp    7baa1 <_int_free+0x911>
 0.00 :        7bac4:       lea    0xf14bd(%rip),%rsi        # 16cf88 <__PRETTY_FUNCTION__.9328+0x1b60>
 0.00 :        7bacb:       jmpq   7b9a9 <_int_free+0x819>
 0.00 :        7bad0:       mov    0x28(%rdx),%rsi
 0.00 :        7bad4:       cmp    %r12,%rsi
 0.00 :        7bad7:       jne    7bc06 <_int_free+0xa76>
 0.00 :        7badd:       mov    0x28(%r12),%rcx
 0.00 :        7bae2:       cmp    0x20(%rcx),%r12
 0.00 :        7bae6:       jne    7bbe7 <_int_free+0xa57>
 0.00 :        7baec:       cmpq   $0x0,0x20(%rax)
 0.00 :        7baf1:       je     7bb4b <_int_free+0x9bb>
 0.00 :        7baf3:       mov    %rcx,0x28(%rdx)
 0.00 :        7baf7:       mov    0x28(%r12),%rax
 0.00 :        7bafc:       mov    %rdx,0x20(%rax)
 0.00 :        7bb00:       jmpq   7b3ba <_int_free+0x22a>
 0.00 :        7bb05:       mov    0x32b6ad(%rip),%edi        # 3a71b8 <check_action>
 0.00 :        7bb0b:       lea    0xed281(%rip),%rsi        # 168d93 <__PRETTY_FUNCTION__.11753+0xf6>
 0.00 :        7bb12:       mov    %r12,%rdx
 0.00 :        7bb15:       callq  7aa50 <malloc_printerr>
 0.00 :        7bb1a:       jmpq   7b3ba <_int_free+0x22a>
 0.00 :        7bb1f:       cmp    %rbx,0x28(%rdx)
 0.00 :        7bb23:       jne    7bbc8 <_int_free+0xa38>
 0.00 :        7bb29:       mov    0x28(%rbx),%rcx
 0.00 :        7bb2d:       cmp    0x20(%rcx),%rbx
 0.00 :        7bb31:       jne    7bba9 <_int_free+0xa19>
 0.00 :        7bb33:       cmpq   $0x0,0x20(%rax)
 0.00 :        7bb38:       je     7bb6d <_int_free+0x9dd>
 0.00 :        7bb3a:       mov    %rcx,0x28(%rdx)
 0.00 :        7bb3e:       mov    0x28(%rbx),%rax
 0.00 :        7bb42:       mov    %rdx,0x20(%rax)
```

```
 0.00 :        7bb46:       jmpq   7b365 <_int_free+0x1d5>
 0.00 :        7bb4b:       cmp    %r12,%rdx
 0.00 :        7bb4e:       je     7bb8f <_int_free+0x9ff>
 0.00 :        7bb50:       mov    %rdx,0x20(%rax)
 0.00 :        7bb54:       mov    0x20(%rsi),%rdx
 0.00 :        7bb58:       mov    %rcx,0x28(%rax)
 0.00 :        7bb5c:       mov    %rax,0x28(%rdx)
 0.00 :        7bb60:       mov    0x28(%rsi),%rdx
 0.00 :        7bb64:       mov    %rax,0x20(%rdx)
 0.00 :        7bb68:       jmpq   7b3ba <_int_free+0x22a>
 0.00 :        7bb6d:       cmp    %rbx,%rdx
 0.00 :        7bb70:       je     7bb9c <_int_free+0xa0c>
 0.00 :        7bb72:       mov    %rdx,0x20(%rax)
 0.00 :        7bb76:       mov    0x20(%rbx),%rdx
 0.00 :        7bb7a:       mov    %rcx,0x28(%rax)
 0.00 :        7bb7e:       mov    %rax,0x28(%rdx)
 0.00 :        7bb82:       mov    0x28(%rbx),%rdx
 0.00 :        7bb86:       mov    %rax,0x20(%rdx)
 0.00 :        7bb8a:       jmpq   7b365 <_int_free+0x1d5>
 0.00 :        7bb8f:       mov    %rax,0x28(%rax)
 0.00 :        7bb93:       mov    %rax,0x20(%rax)
 0.00 :        7bb97:       jmpq   7b3ba <_int_free+0x22a>
 0.00 :        7bb9c:       mov    %rax,0x28(%rax)
 0.00 :        7bba0:       mov    %rax,0x20(%rax)
 0.00 :        7bba4:       jmpq   7b365 <_int_free+0x1d5>
 0.00 :        7bba9:       lea    0xed4c6(%rip),%rcx        # 169076 <__func__.11466>
 0.00 :        7bbb0:       lea    0xed19e(%rip),%rsi        # 168d55 <__PRETTY_FUNCTION__.11753+0xb8>
 0.00 :        7bbb7:       lea    0xf12aa(%rip),%rdi        # 16ce68 <__PRETTY_FUNCTION__.9328+0x1a40>
 0.00 :        7bbbe:       mov    $0xf3f,%edx
 0.00 :        7bbc3:       callq  7a110 <__malloc_assert>
 0.00 :        7bbc8:       lea    0xed4a7(%rip),%rcx        # 169076 <__func__.11466>
 0.00 :        7bbcf:       lea    0xed17f(%rip),%rsi        # 168d55 <__PRETTY_FUNCTION__.11753+0xb8>
 0.00 :        7bbd6:       lea    0xf1263(%rip),%rdi        # 16ce40 <__PRETTY_FUNCTION__.9328+0x1a18>
 0.00 :        7bbdd:       mov    $0xf3f,%edx
 0.00 :        7bbe2:       callq  7a110 <__malloc_assert>
 0.00 :        7bbe7:       lea    0xed488(%rip),%rcx        # 169076 <__func__.11466>
 0.00 :        7bbee:       lea    0xed160(%rip),%rsi        # 168d55 <__PRETTY_FUNCTION__.11753+0xb8>
 0.00 :        7bbf5:       lea    0xf12cc(%rip),%rdi        # 16cec8 <__PRETTY_FUNCTION__.9328+0x1aa0>
 0.00 :        7bbfc:       mov    $0xf48,%edx
 0.00 :        7bc01:       callq  7a110 <__malloc_assert>
 0.00 :        7bc06:       lea    0xed469(%rip),%rcx        # 169076 <__func__.11466>
 0.00 :        7bc0d:       lea    0xed141(%rip),%rsi        # 168d55 <__PRETTY_FUNCTION__.11753+0xb8>
 0.00 :        7bc14:       lea    0xf1275(%rip),%rdi        # 16ce90 <__PRETTY_FUNCTION__.9328+0x1a68>
 0.00 :        7bc1b:       mov    $0xf48,%edx
 0.00 :        7bc20:       callq  7a110 <__malloc_assert>
 0.00 :        7bc25:       mov    0x32b58d(%rip),%edi        # 3a71b8 <check_action>
 0.00 :        7bc2b:       lea    0xed161(%rip),%rsi        # 168d93 <__PRETTY_FUNCTION__.11753+0xf6>
 0.00 :        7bc32:       mov    %rbx,%rdx
 0.00 :        7bc35:       callq  7aa50 <malloc_printerr>
 0.00 :        7bc3a:       jmpq   7b365 <_int_free+0x1d5>
 0.00 :        7bc3f:       lea    0xf1302(%rip),%rsi        # 16cf48 <__PRETTY_FUNCTION__.9328+0x1b20>
 0.00 :        7bc46:       jmpq   7b9a9 <_int_free+0x819>
 0.00 :        7bc4b:       mov    0x8(%rax),%rdx
50.00 :        7bc4f:       and    $0xfffffffffffffff8,%rdx
 0.00 :        7bc53:       add    %rdx,%rax
 0.00 :        7bc56:       cmp    %rax,%r12
 0.00 :        7bc59:       jb     7b2e2 <_int_free+0x152>
 0.00 :        7bc5f:       lea    0xf1302(%rip),%rsi        # 16cf68 <__PRETTY_FUNCTION__.9328+0x1b40>
 0.00 :        7bc66:       jmpq   7b9a9 <_int_free+0x819>
 0.00 :        7bc6b:       lea    0xf12ae(%rip),%rsi        # 16cf20 <__PRETTY_FUNCTION__.9328+0x1af8>
 0.00 :        7bc72:       jmpq   7b9a9 <_int_free+0x819>
 0.00 :        7bc77:       lea    -0x10(%rbp),%rdx
 0.00 :        7bc7b:       lea    0x10(%rbx),%rdi
 0.00 :        7bc7f:       movzbl %al,%esi
```

```
    0.00 :          7bc82:       callq  86cd0 <__GI_memset>
    0.00 :          7bc87:       jmpq   7b31b <_int_free+0x18b>
    0.00 :          7bc8c:       cmp    %rdx,%rbx
    0.00 :          7bc8f:       je     7bcec <_int_free+0xb5c>
    0.00 :          7bc91:       mov    %rdx,0x20(%rax)
    0.00 :          7bc95:       mov    0x20(%rbx),%rdx
    0.00 :          7bc99:       mov    %rcx,0x28(%rax)
    0.00 :          7bc9d:       mov    %rax,0x28(%rdx)
    0.00 :          7bca1:       mov    0x28(%rbx),%rdx
    0.00 :          7bca5:       mov    %rax,0x20(%rdx)
    0.00 :          7bca9:       jmpq   7b514 <_int_free+0x384>
    0.00 :          7bcae:       lea    0xed3cb(%rip),%rcx        # 169080 <__func__.10900>
    0.00 :          7bcb5:       lea    0xed16e(%rip),%rsi        # 168e2a <__PRETTY_FUNCTION__.11753+0x18d>
    0.00 :          7bcbc:       lea    0xf11a5(%rip),%rdi        # 16ce68 <__PRETTY_FUNCTION__.9328+0x1a40>
    0.00 :          7bcc3:       mov    $0x2ae,%edx
    0.00 :          7bcc8:       callq  7a110 <__malloc_assert>
    0.00 :          7bccd:       lea    0xed3ac(%rip),%rcx        # 169080 <__func__.10900>
    0.00 :          7bcd4:       lea    0xed14f(%rip),%rsi        # 168e2a <__PRETTY_FUNCTION__.11753+0x18d>
    0.00 :          7bcdb:       lea    0xf115e(%rip),%rdi        # 16ce40 <__PRETTY_FUNCTION__.9328+0x1a18>
    0.00 :          7bce2:       mov    $0x2ae,%edx
    0.00 :          7bce7:       callq  7a110 <__malloc_assert>
    0.00 :          7bcec:       mov    %rax,0x28(%rax)
    0.00 :          7bcf0:       mov    %rax,0x20(%rax)
    0.00 :          7bcf4:       jmpq   7b514 <_int_free+0x384>
 Percent |       Source code & Disassembly of libc-2.17.so
-----------------------------------------------
         :
         :
         :
         :       Disassembly of section .text:
         :
         :       000000000007bd00 <_int_malloc>:
    0.00 :          7bd00:       push   %r15
    0.00 :          7bd02:       push   %r14
    0.00 :          7bd04:       push   %r13
    0.00 :          7bd06:       push   %r12
    0.00 :          7bd08:       mov    %rsi,%r12
    0.00 :          7bd0b:       push   %rbp
    0.00 :          7bd0c:       push   %rbx
    0.00 :          7bd0d:       sub    $0x68,%rsp
    0.00 :          7bd11:       cmp    $0xfffffffffffffbf,%rsi
    0.00 :          7bd15:       ja     7c3fe <_int_malloc+0x6fe>
    0.00 :          7bd1b:       lea    0x17(%rsi),%rax
    0.00 :          7bd1f:       mov    $0x20,%ebp
    0.00 :          7bd24:       mov    %rdi,%rbx
    0.00 :          7bd27:       mov    %rax,%rdx
    0.00 :          7bd2a:       and    $0xfffffffffffffff0,%rdx
    0.00 :          7bd2e:       cmp    $0x20,%rax
    0.00 :          7bd32:       cmovae %rdx,%rbp
    0.00 :          7bd36:       cmp    0x32db23(%rip),%rbp        # 3a9860 <global_max_fast>
    0.00 :          7bd3d:       ja     7bde8 <_int_malloc+0xe8>
    0.00 :          7bd43:       mov    %ebp,%edi
    0.00 :          7bd45:       shr    $0x4,%edi
    0.00 :          7bd48:       sub    $0x2,%edi
    0.00 :          7bd4b:       mov    %edi,%eax
    0.00 :          7bd4d:       mov    0x8(%rbx,%rax,8),%r13
    0.00 :          7bd52:       lea    (%rbx,%rax,8),%rsi
    0.00 :          7bd56:       lea    0x8(%rsi),%rcx
    0.00 :          7bd5a:       test   %r13,%r13
    0.00 :          7bd5d:       je     7bde8 <_int_malloc+0xe8>
    0.00 :          7bd63:       mov    0x10(%r13),%r8
    0.00 :          7bd67:       mov    %r13,%rax
    0.00 :          7bd6a:       cmpl   $0x0,%fs:0x18
    0.00 :          7bd73:       je     7bd76 <_int_malloc+0x76>
```

```
 0.00 :         7bd75:       lock cmpxchg %r8,0x8(%rsi)
 0.00 :         7bd7b:       cmp    %rax,%r13
 0.00 :         7bd7e:       mov    %rax,%rdx
 0.00 :         7bd81:       jne    7bd8b <_int_malloc+0x8b>
 0.00 :         7bd83:       jmp    7bdaf <_int_malloc+0xaf>
 0.00 :         7bd85:       nopl   (%rax)
 0.00 :         7bd88:       mov    %rax,%rdx
 0.00 :         7bd8b:       test   %rdx,%rdx
 0.00 :         7bd8e:       je     7bde8 <_int_malloc+0xe8>
 0.00 :         7bd90:       mov    0x10(%rdx),%rsi
 0.00 :         7bd94:       mov    %rdx,%rax
 0.00 :         7bd97:       cmpl   $0x0,%fs:0x18
 0.00 :         7bda0:       je     7bda3 <_int_malloc+0xa3>
 0.00 :         7bda2:       lock cmpxchg %rsi,(%rcx)
 0.00 :         7bda7:       cmp    %rdx,%rax
 0.00 :         7bdaa:       jne    7bd88 <_int_malloc+0x88>
 0.00 :         7bdac:       mov    %rdx,%r13
 0.00 :         7bdaf:       mov    0x8(%r13),%eax
 0.00 :         7bdb3:       shr    $0x4,%eax
 0.00 :         7bdb6:       sub    $0x2,%eax
 0.00 :         7bdb9:       cmp    %eax,%edi
 0.00 :         7bdbb:       jne    7ce26 <_int_malloc+0x1126>
 0.00 :         7bdc1:       mov    0x32daa1(%rip),%eax        # 3a9868 <perturb_byte>
 0.00 :         7bdc7:       add    $0x10,%r13
 0.00 :         7bdcb:       test   %eax,%eax
 0.00 :         7bdcd:       jne    7c14f <_int_malloc+0x44f>
 0.00 :         7bdd3:       add    $0x68,%rsp
 0.00 :         7bdd7:       mov    %r13,%rax
 0.00 :         7bdda:       pop    %rbx
 0.00 :         7bddb:       pop    %rbp
 0.00 :         7bddc:       pop    %r12
 0.00 :         7bdde:       pop    %r13
 0.00 :         7bde0:       pop    %r14
 0.00 :         7bde2:       pop    %r15
 0.00 :         7bde4:       retq
 0.00 :         7bde5:       nopl   (%rax)
 0.00 :         7bde8:       cmp    $0x3ff,%rbp
 0.00 :         7bdef:       ja     7be3f <_int_malloc+0x13f>
 0.00 :         7bdf1:       mov    %ebp,%r9d
 0.00 :         7bdf4:       shr    $0x4,%r9d
 0.00 :         7bdf8:       lea    -0x2(%r9,%r9,1),%eax
 0.00 :         7bdfd:       lea    0x58(%rbx,%rax,8),%rax
 0.00 :         7be02:       mov    0x18(%rax),%r13
 0.00 :         7be06:       cmp    %rax,%r13
 0.00 :         7be09:       je     7be6c <_int_malloc+0x16c>
 0.00 :         7be0b:       test   %r13,%r13
 0.00 :         7be0e:       je     7be5a <_int_malloc+0x15a>
 0.00 :         7be10:       mov    0x18(%r13),%rdx
 0.00 :         7be14:       cmp    0x10(%rdx),%r13
 0.00 :         7be18:       jne    7cebe <_int_malloc+0x11be>
 0.00 :         7be1e:       mov    %rdx,0x18(%rax)
 0.00 :         7be22:       mov    %rax,0x10(%rdx)
 0.00 :         7be26:       lea    0x32b813(%rip),%rax        # 3a7640 <main_arena>
 0.00 :         7be2d:       orq    $0x1,0x8(%r13,%rbp,1)
 0.00 :         7be33:       cmp    %rax,%rbx
 0.00 :         7be36:       je     7bdc1 <_int_malloc+0xc1>
 0.00 :         7be38:       orq    $0x4,0x8(%r13)
 0.00 :         7be3d:       jmp    7bdc1 <_int_malloc+0xc1>
 0.00 :         7be3f:       mov    %rbp,%r9
 0.00 :         7be42:       shr    $0x6,%r9
 0.00 :         7be46:       cmp    $0x30,%r9
 0.00 :         7be4a:       ja     7c3e8 <_int_malloc+0x6e8>
 0.00 :         7be50:       add    $0x30,%r9d
 0.00 :         7be54:       testb  $0x1,0x4(%rbx)
```

```
 0.00 :        7be58:       jne    7be6c <_int_malloc+0x16c>
 0.00 :        7be5a:       mov    %rbx,%rdi
 0.00 :        7be5d:       mov    %r9d,0x8(%rsp)
 0.00 :        7be62:       callq  7ab30 <malloc_consolidate>
 0.00 :        7be67:       mov    0x8(%rsp),%r9d
 0.00 :        7be6c:       mov    %ebp,%eax
 0.00 :        7be6e:       mov    %rbp,%r8
 0.00 :        7be71:       lea    0x20(%rbp),%rdx
 0.00 :        7be75:       shr    $0x4,%eax
 0.00 :        7be78:       shr    $0x6,%r8
 0.00 :        7be7c:       lea    0x58(%rbx),%r14
 0.00 :        7be80:       mov    %eax,0x48(%rsp)
 0.00 :        7be84:       lea    0x30(%r8),%eax
 0.00 :        7be88:       mov    %rdx,0x20(%rsp)
 0.00 :        7be8d:       mov    %r8,%r15
 0.00 :        7be90:       mov    %eax,0x4c(%rsp)
 0.00 :        7be94:       mov    %rbp,%rax
 0.00 :        7be97:       shr    $0x9,%rax
 0.00 :        7be9b:       mov    %rax,0x30(%rsp)
 0.00 :        7bea0:       add    $0x5b,%eax
 0.00 :        7bea3:       mov    %eax,0x50(%rsp)
 0.00 :        7bea7:       mov    %rbp,%rax
 0.00 :        7beaa:       shr    $0xc,%rax
 0.00 :        7beae:       mov    %rax,0x28(%rsp)
 0.00 :        7beb3:       add    $0x6e,%eax
 0.00 :        7beb6:       mov    %eax,0x54(%rsp)
 0.00 :        7beba:       mov    %rbp,%rax
 0.00 :        7bebd:       shr    $0xf,%rax
 0.00 :        7bec1:       mov    %rax,0x38(%rsp)
 0.00 :        7bec6:       add    $0x77,%eax
 0.00 :        7bec9:       mov    %eax,0x58(%rsp)
 0.00 :        7becd:       mov    %rbp,%rax
 0.00 :        7bed0:       shr    $0x12,%rax
 0.00 :        7bed4:       mov    %rax,0x40(%rsp)
 0.00 :        7bed9:       add    $0x7c,%eax
 0.00 :        7bedc:       mov    %eax,0x5c(%rsp)
 0.00 :        7bee0:       mov    $0x2710,%r8d
 0.00 :        7bee6:       jmp    7bf34 <_int_malloc+0x234>
 0.00 :        7bee8:       nopl   0x0(%rax,%rax,1)
 0.00 :        7bef0:       mov    %esi,%ecx
 0.00 :        7bef2:       shr    $0x4,%ecx
 0.00 :        7bef5:       lea    -0x2(%rcx,%rcx,1),%eax
 0.00 :        7bef9:       cltq
 0.00 :        7befb:       lea    0x58(%rbx,%rax,8),%rdx
 0.00 :        7bf00:       mov    0x10(%rdx),%rax
 0.00 :        7bf04:       mov    %ecx,%edi
 0.00 :        7bf06:       mov    $0x1,%esi
 0.00 :        7bf0b:       sar    $0x5,%edi
 0.00 :        7bf0e:       shl    %cl,%esi
 0.00 :        7bf10:       movslq %edi,%rdi
 0.00 :        7bf13:       or     %esi,0x858(%rbx,%rdi,4)
 0.00 :        7bf1a:       sub    $0x1,%r8d
 0.00 :        7bf1e:       mov    %rdx,0x18(%r13)
 0.00 :        7bf22:       mov    %rax,0x10(%r13)
 0.00 :        7bf26:       mov    %r13,0x18(%rax)
 0.00 :        7bf2a:       mov    %r13,0x10(%rdx)
 0.00 :        7bf2e:       je     7c1bf <_int_malloc+0x4bf>
 0.00 :        7bf34:       mov    0x70(%rbx),%r13
50.00 :        7bf38:       cmp    %r14,%r13
 0.00 :        7bf3b:       je     7c1bf <_int_malloc+0x4bf>
 0.00 :        7bf41:       mov    0x8(%r13),%rsi
 0.00 :        7bf45:       mov    0x18(%r13),%rax
 0.00 :        7bf49:       cmp    $0x10,%rsi
 0.00 :        7bf4d:       jbe    7c5c3 <_int_malloc+0x8c3>
```

```
 0.00 :        7bf53:       cmp    0x878(%rbx),%rsi
 0.00 :        7bf5a:       ja     7c5c3 <_int_malloc+0x8c3>
 0.00 :        7bf60:       and    $0xfffffffffffffff8,%rsi
 0.00 :        7bf64:       cmp    $0x3ff,%rbp
 0.00 :        7bf6b:       ja     7bf76 <_int_malloc+0x276>
 0.00 :        7bf6d:       cmp    %rax,%r14
 0.00 :        7bf70:       je     7c0c0 <_int_malloc+0x3c0>
 0.00 :        7bf76:       cmp    %rsi,%rbp
 0.00 :        7bf79:       mov    %rax,0x70(%rbx)
 0.00 :        7bf7d:       mov    %r14,0x10(%rax)
50.00 :        7bf81:       je     7c300 <_int_malloc+0x600>
 0.00 :        7bf87:       cmp    $0x3ff,%rsi
 0.00 :        7bf8e:       jbe    7bef0 <_int_malloc+0x1f0>
 0.00 :        7bf94:       mov    %rsi,%rax
 0.00 :        7bf97:       shr    $0x6,%rax
 0.00 :        7bf9b:       cmp    $0x30,%rax
 0.00 :        7bf9f:       ja     7bff8 <_int_malloc+0x2f8>
 0.00 :        7bfa1:       lea    0x30(%rax),%ecx
 0.00 :        7bfa4:       lea    0x5e(%rax,%rax,1),%eax
 0.00 :        7bfa8:       cltq
 0.00 :        7bfaa:       sub    $0x2,%rax
 0.00 :        7bfae:       lea    0x68(%rbx,%rax,8),%rdi
 0.00 :        7bfb3:       mov    0x10(%rdi),%rax
 0.00 :        7bfb7:       cmp    %rdi,%rax
 0.00 :        7bfba:       je     7c027 <_int_malloc+0x327>
 0.00 :        7bfbc:       mov    0x18(%rdi),%rdx
 0.00 :        7bfc0:       or     $0x1,%rsi
 0.00 :        7bfc4:       mov    0x8(%rdx),%r10
 0.00 :        7bfc8:       test   $0x4,%r10b
 0.00 :        7bfcc:       jne    7ce32 <_int_malloc+0x1132>
 0.00 :        7bfd2:       cmp    %r10,%rsi
 0.00 :        7bfd5:       jae    7c040 <_int_malloc+0x340>
 0.00 :        7bfd7:       mov    0x28(%rax),%rsi
 0.00 :        7bfdb:       mov    %rax,0x20(%r13)
 0.00 :        7bfdf:       mov    %rsi,0x28(%r13)
 0.00 :        7bfe3:       mov    %r13,0x20(%rsi)
 0.00 :        7bfe7:       mov    %r13,0x28(%rax)
 0.00 :        7bfeb:       mov    %rdi,%rax
 0.00 :        7bfee:       jmpq   7bf04 <_int_malloc+0x204>
 0.00 :        7bff3:       nopl   0x0(%rax,%rax,1)
 0.00 :        7bff8:       mov    %rsi,%rax
 0.00 :        7bffb:       shr    $0x9,%rax
 0.00 :        7bfff:       cmp    $0x14,%rax
 0.00 :        7c003:       ja     7c090 <_int_malloc+0x390>
 0.00 :        7c009:       lea    0x5b(%rax),%ecx
 0.00 :        7c00c:       lea    0xb4(%rax,%rax,1),%eax
 0.00 :        7c013:       cltq
 0.00 :        7c015:       sub    $0x2,%rax
 0.00 :        7c019:       lea    0x68(%rbx,%rax,8),%rdi
 0.00 :        7c01e:       mov    0x10(%rdi),%rax
 0.00 :        7c022:       cmp    %rdi,%rax
 0.00 :        7c025:       jne    7bfbc <_int_malloc+0x2bc>
 0.00 :        7c027:       mov    %r13,0x28(%r13)
 0.00 :        7c02b:       mov    %r13,0x20(%r13)
 0.00 :        7c02f:       mov    %rax,%rdx
 0.00 :        7c032:       jmpq   7bf04 <_int_malloc+0x204>
 0.00 :        7c037:       nopw   0x0(%rax,%rax,1)
 0.00 :        7c040:       mov    0x8(%rax),%rdx
 0.00 :        7c044:       test   $0x4,%dl
 0.00 :        7c047:       je     7c061 <_int_malloc+0x361>
 0.00 :        7c049:       jmpq   7cfe4 <_int_malloc+0x12e4>
 0.00 :        7c04e:       xchg   %ax,%ax
 0.00 :        7c050:       mov    0x20(%rax),%rax
 0.00 :        7c054:       mov    0x8(%rax),%rdx
```

```
  0.00 :         7c058:       test   $0x4,%dl
  0.00 :         7c05b:       jne    7c568 <_int_malloc+0x868>
  0.00 :         7c061:       cmp    %rdx,%rsi
  0.00 :         7c064:       jb     7c050 <_int_malloc+0x350>
  0.00 :         7c066:       je     7c190 <_int_malloc+0x490>
  0.00 :         7c06c:       mov    0x28(%rax),%rdx
  0.00 :         7c070:       mov    %rax,0x20(%r13)
  0.00 :         7c074:       mov    %rdx,0x28(%r13)
  0.00 :         7c078:       mov    %r13,0x28(%rax)
  0.00 :         7c07c:       mov    0x28(%r13),%rdx
  0.00 :         7c080:       mov    %r13,0x20(%rdx)
  0.00 :         7c084:       mov    0x18(%rax),%rdx
  0.00 :         7c088:       jmpq   7bf04 <_int_malloc+0x204>
  0.00 :         7c08d:       nopl   (%rax)
  0.00 :         7c090:       mov    %rsi,%rax
  0.00 :         7c093:       shr    $0xc,%rax
  0.00 :         7c097:       cmp    $0xa,%rax
  0.00 :         7c09b:       ja     7c168 <_int_malloc+0x468>
  0.00 :         7c0a1:       lea    0x6e(%rax),%ecx
  0.00 :         7c0a4:       lea    0xda(%rax,%rax,1),%eax
  0.00 :         7c0ab:       cltq
  0.00 :         7c0ad:       sub    $0x2,%rax
  0.00 :         7c0b1:       jmpq   7bfae <_int_malloc+0x2ae>
  0.00 :         7c0b6:       nopw   %cs:0x0(%rax,%rax,1)
  0.00 :         7c0c0:       mov    0x60(%rbx),%rcx
  0.00 :         7c0c4:       cmp    %r13,%rcx
  0.00 :         7c0c7:       jne    7bf76 <_int_malloc+0x276>
  0.00 :         7c0cd:       cmp    0x20(%rsp),%rsi
  0.00 :         7c0d2:       jbe    7bf76 <_int_malloc+0x276>
  0.00 :         7c0d8:       mov    %rsi,%rdx
  0.00 :         7c0db:       add    %rbp,%r13
  0.00 :         7c0de:       sub    %rbp,%rdx
  0.00 :         7c0e1:       mov    %r13,0x60(%rbx)
  0.00 :         7c0e5:       mov    %r13,0x68(%rbx)
  0.00 :         7c0e9:       cmp    $0x3ff,%rdx
  0.00 :         7c0f0:       mov    %r13,0x70(%rbx)
  0.00 :         7c0f4:       mov    %r14,0x10(%r13)
  0.00 :         7c0f8:       mov    %r14,0x18(%r13)
  0.00 :         7c0fc:       jbe    7c10e <_int_malloc+0x40e>
  0.00 :         7c0fe:       movq   $0x0,0x20(%r13)
  0.00 :         7c106:       movq   $0x0,0x28(%r13)
  0.00 :         7c10e:       lea    0x32b52b(%rip),%rax        # 3a7640 <main_arena>
  0.00 :         7c115:       mov    %rdx,0x0(%r13,%rdx,1)
  0.00 :         7c11a:       cmp    %rax,%rbx
  0.00 :         7c11d:       setne  %al
  0.00 :         7c120:       or     $0x1,%rbp
  0.00 :         7c124:       movzbl %al,%eax
  0.00 :         7c127:       shl    $0x2,%rax
  0.00 :         7c12b:       or     %rax,%rbp
  0.00 :         7c12e:       mov    %rdx,%rax
  0.00 :         7c131:       or     $0x1,%rax
  0.00 :         7c135:       mov    %rbp,0x8(%rcx)
  0.00 :         7c139:       mov    %rax,0x8(%r13)
  0.00 :         7c13d:       mov    0x32d725(%rip),%eax        # 3a9868 <perturb_byte>
  0.00 :         7c143:       lea    0x10(%rcx),%r13
  0.00 :         7c147:       test   %eax,%eax
  0.00 :         7c149:       je     7bdd3 <_int_malloc+0xd3>
  0.00 :         7c14f:       not    %eax
  0.00 :         7c151:       mov    %r12,%rdx
  0.00 :         7c154:       mov    %r13,%rdi
  0.00 :         7c157:       movzbl %al,%esi
  0.00 :         7c15a:       callq  86cd0 <__GI_memset>
  0.00 :         7c15f:       jmpq   7bdd3 <_int_malloc+0xd3>
  0.00 :         7c164:       nopl   0x0(%rax)
```

```
 0.00 :        7c168:       mov    %rsi,%rax
 0.00 :        7c16b:       shr    $0xf,%rax
 0.00 :        7c16f:       cmp    $0x4,%rax
 0.00 :        7c173:       ja     7c199 <_int_malloc+0x499>
 0.00 :        7c175:       lea    0x77(%rax),%ecx
 0.00 :        7c178:       lea    0xec(%rax,%rax,1),%eax
 0.00 :        7c17f:       cltq
 0.00 :        7c181:       sub    $0x2,%rax
 0.00 :        7c185:       jmpq   7bfae <_int_malloc+0x2ae>
 0.00 :        7c18a:       nopw   0x0(%rax,%rax,1)
 0.00 :        7c190:       mov    0x10(%rax),%rax
 0.00 :        7c194:       jmpq   7c084 <_int_malloc+0x384>
 0.00 :        7c199:       mov    %rsi,%rax
 0.00 :        7c19c:       shr    $0x12,%rax
 0.00 :        7c1a0:       cmp    $0x2,%rax
 0.00 :        7c1a4:       ja     7c320 <_int_malloc+0x620>
 0.00 :        7c1aa:       lea    0x7c(%rax),%ecx
 0.00 :        7c1ad:       lea    0xf6(%rax,%rax,1),%eax
 0.00 :        7c1b4:       cltq
 0.00 :        7c1b6:       sub    $0x2,%rax
 0.00 :        7c1ba:       jmpq   7bfae <_int_malloc+0x2ae>
 0.00 :        7c1bf:       cmp    $0x3ff,%rbp
 0.00 :        7c1c6:       ja     7c32f <_int_malloc+0x62f>
 0.00 :        7c1cc:       lea    0x1(%r9),%ecx
 0.00 :        7c1d0:       lea    (%r9,%r9,1),%eax
 0.00 :        7c1d4:       mov    %ecx,%edi
 0.00 :        7c1d6:       lea    0x58(%rbx,%rax,8),%rdx
 0.00 :        7c1db:       mov    $0x1,%eax
 0.00 :        7c1e0:       shr    $0x5,%edi
 0.00 :        7c1e3:       shl    %cl,%eax
 0.00 :        7c1e5:       mov    %edi,%r8d
 0.00 :        7c1e8:       mov    0x858(%rbx,%r8,4),%esi
 0.00 :        7c1f0:       test   %eax,%eax
 0.00 :        7c1f2:       je     7c200 <_int_malloc+0x500>
 0.00 :        7c1f4:       cmp    %esi,%eax
 0.00 :        7c1f6:       jbe    7c23c <_int_malloc+0x53c>
 0.00 :        7c1f8:       nopl   0x0(%rax,%rax,1)
 0.00 :        7c200:       add    $0x1,%edi
 0.00 :        7c203:       cmp    $0x3,%edi
 0.00 :        7c206:       ja     7c263 <_int_malloc+0x563>
 0.00 :        7c208:       mov    %edi,%r8d
 0.00 :        7c20b:       mov    0x858(%rbx,%r8,4),%esi
 0.00 :        7c213:       test   %esi,%esi
 0.00 :        7c215:       je     7c200 <_int_malloc+0x500>
 0.00 :        7c217:       mov    %edi,%eax
 0.00 :        7c219:       shl    $0x6,%eax
 0.00 :        7c21c:       sub    $0x2,%eax
 0.00 :        7c21f:       lea    0x58(%rbx,%rax,8),%rdx
 0.00 :        7c224:       mov    $0x1,%eax
 0.00 :        7c229:       test   %esi,%eax
 0.00 :        7c22b:       jne    7c240 <_int_malloc+0x540>
 0.00 :        7c22d:       nopl   (%rax)
 0.00 :        7c230:       add    $0x10,%rdx
 0.00 :        7c234:       add    %eax,%eax
 0.00 :        7c236:       je     7c587 <_int_malloc+0x887>
 0.00 :        7c23c:       test   %esi,%eax
 0.00 :        7c23e:       je     7c230 <_int_malloc+0x530>
 0.00 :        7c240:       mov    0x18(%rdx),%r13
 0.00 :        7c244:       cmp    %r13,%rdx
 0.00 :        7c247:       jne    7c43f <_int_malloc+0x73f>
 0.00 :        7c24d:       mov    %eax,%ecx
 0.00 :        7c24f:       add    $0x10,%rdx
 0.00 :        7c253:       add    %eax,%eax
 0.00 :        7c255:       not    %ecx
```

```
 0.00 :        7c257:       and    %ecx,%esi
 0.00 :        7c259:       mov    %esi,0x858(%rbx,%r8,4)
 0.00 :        7c261:       jmp    7c1f0 <_int_malloc+0x4f0>
 0.00 :        7c263:       mov    0x58(%rbx),%rdx
 0.00 :        7c267:       mov    0x8(%rdx),%rax
 0.00 :        7c26b:       mov    %rax,%r10
 0.00 :        7c26e:       and    $0xfffffffffffffff8,%r10
 0.00 :        7c272:       cmp    0x20(%rsp),%r10
 0.00 :        7c277:       jae    7c600 <_int_malloc+0x900>
 0.00 :        7c27d:       testb  $0x1,0x4(%rbx)
 0.00 :        7c281:       jne    7c650 <_int_malloc+0x950>
 0.00 :        7c287:       mov    %rbx,%rdi
 0.00 :        7c28a:       callq  7ab30 <malloc_consolidate>
 0.00 :        7c28f:       cmp    $0x3ff,%rbp
 0.00 :        7c296:       mov    0x48(%rsp),%r9d
 0.00 :        7c29b:       jbe    7bee0 <_int_malloc+0x1e0>
 0.00 :        7c2a1:       cmp    $0x30,%r15
 0.00 :        7c2a5:       mov    0x4c(%rsp),%r9d
 0.00 :        7c2aa:       jbe    7bee0 <_int_malloc+0x1e0>
 0.00 :        7c2b0:       cmpq   $0x14,0x30(%rsp)
 0.00 :        7c2b6:       mov    0x50(%rsp),%r9d
 0.00 :        7c2bb:       jbe    7bee0 <_int_malloc+0x1e0>
 0.00 :        7c2c1:       cmpq   $0xa,0x28(%rsp)
 0.00 :        7c2c7:       mov    0x54(%rsp),%r9d
 0.00 :        7c2cc:       jbe    7bee0 <_int_malloc+0x1e0>
 0.00 :        7c2d2:       cmpq   $0x4,0x38(%rsp)
 0.00 :        7c2d8:       mov    0x58(%rsp),%r9d
 0.00 :        7c2dd:       jbe    7bee0 <_int_malloc+0x1e0>
 0.00 :        7c2e3:       cmpq   $0x3,0x40(%rsp)
 0.00 :        7c2e9:       mov    $0x7e,%r9d
 0.00 :        7c2ef:       cmovb  0x5c(%rsp),%r9d
 0.00 :        7c2f5:       jmpq   7bee0 <_int_malloc+0x1e0>
 0.00 :        7c2fa:       nopw   0x0(%rax,%rax,1)
 0.00 :        7c300:       lea    0x32b339(%rip),%rax        # 3a7640 <main_arena>
 0.00 :        7c307:       orq    $0x1,0x8(%r13,%rbp,1)
 0.00 :        7c30d:       cmp    %rax,%rbx
 0.00 :        7c310:       je     7bdc1 <_int_malloc+0xc1>
 0.00 :        7c316:       orq    $0x4,0x8(%r13)
 0.00 :        7c31b:       jmpq   7bdc1 <_int_malloc+0xc1>
 0.00 :        7c320:       mov    $0xf8,%eax
 0.00 :        7c325:       mov    $0x7e,%ecx
 0.00 :        7c32a:       jmpq   7bfae <_int_malloc+0x2ae>
 0.00 :        7c32f:       lea    -0x2(%r9,%r9,1),%eax
 0.00 :        7c334:       lea    0x58(%rbx,%rax,8),%rax
 0.00 :        7c339:       mov    0x10(%rax),%rdx
 0.00 :        7c33d:       cmp    %rax,%rdx
 0.00 :        7c340:       je     7c1cc <_int_malloc+0x4cc>
 0.00 :        7c346:       cmp    0x8(%rdx),%rbp
 0.00 :        7c34a:       ja     7c1cc <_int_malloc+0x4cc>
 0.00 :        7c350:       mov    0x28(%rdx),%r13
 0.00 :        7c354:       jmp    7c35a <_int_malloc+0x65a>
 0.00 :        7c356:       mov    0x28(%r13),%r13
 0.00 :        7c35a:       mov    0x8(%r13),%rdx
 0.00 :        7c35e:       mov    %rdx,%r15
 0.00 :        7c361:       and    $0xfffffffffffffff8,%r15
 0.00 :        7c365:       cmp    %r15,%rbp
 0.00 :        7c368:       ja     7c356 <_int_malloc+0x656>
 0.00 :        7c36a:       cmp    %r13,0x18(%rax)
 0.00 :        7c36e:       mov    0x10(%r13),%rax
 0.00 :        7c372:       je     7c381 <_int_malloc+0x681>
 0.00 :        7c374:       cmp    0x8(%rax),%rdx
 0.00 :        7c378:       jne    7c381 <_int_malloc+0x681>
 0.00 :        7c37a:       mov    %rax,%r13
 0.00 :        7c37d:       mov    0x10(%rax),%rax
```

```
  0.00 :        7c381:       mov    %r15,%rcx
  0.00 :        7c384:       mov    0x18(%r13),%rdx
  0.00 :        7c388:       sub    %rbp,%rcx
  0.00 :        7c38b:       cmp    0x18(%rax),%r13
  0.00 :        7c38f:       jne    7cdd6 <_int_malloc+0x10d6>
  0.00 :        7c395:       cmp    0x10(%rdx),%r13
  0.00 :        7c399:       jne    7cdd6 <_int_malloc+0x10d6>
  0.00 :        7c39f:       cmpq   $0x3ff,0x8(%r13)
  0.00 :        7c3a7:       mov    %rdx,0x18(%rax)
  0.00 :        7c3ab:       mov    %rax,0x10(%rdx)
  0.00 :        7c3af:       jbe    7c3be <_int_malloc+0x6be>
  0.00 :        7c3b1:       mov    0x20(%r13),%rdx
  0.00 :        7c3b5:       test   %rdx,%rdx
  0.00 :        7c3b8:       jne    7cdfa <_int_malloc+0x10fa>
  0.00 :        7c3be:       cmp    $0x1f,%rcx
  0.00 :        7c3c2:       ja     7cd01 <_int_malloc+0x1001>
  0.00 :        7c3c8:       lea    0x32b271(%rip),%rax        # 3a7640 <main_arena>
  0.00 :        7c3cf:       orq    $0x1,0x8(%r13,%r15,1)
  0.00 :        7c3d5:       cmp    %rax,%rbx
  0.00 :        7c3d8:       jne    7be38 <_int_malloc+0x138>
  0.00 :        7c3de:       jmpq   7bdc1 <_int_malloc+0xc1>
  0.00 :        7c3e3:       nopl   0x0(%rax,%rax,1)
  0.00 :        7c3e8:       mov    %rbp,%r9
  0.00 :        7c3eb:       shr    $0x9,%r9
  0.00 :        7c3ef:       cmp    $0x14,%r9
  0.00 :        7c3f3:       ja     7c414 <_int_malloc+0x714>
  0.00 :        7c3f5:       add    $0x5b,%r9d
  0.00 :        7c3f9:       jmpq   7be54 <_int_malloc+0x154>
  0.00 :        7c3fe:       mov    0x32aa2b(%rip),%rax        # 3a6e30 <_DYNAMIC+0x2f0>
  0.00 :        7c405:       xor    %r13d,%r13d
  0.00 :        7c408:       movl   $0xc,%fs:(%rax)
  0.00 :        7c40f:       jmpq   7bdd3 <_int_malloc+0xd3>
  0.00 :        7c414:       mov    %rbp,%r9
  0.00 :        7c417:       shr    $0xc,%r9
  0.00 :        7c41b:       cmp    $0xa,%r9
  0.00 :        7c41f:       jbe    7c544 <_int_malloc+0x844>
  0.00 :        7c425:       mov    %rbp,%r9
  0.00 :        7c428:       shr    $0xf,%r9
  0.00 :        7c42c:       cmp    $0x4,%r9
  0.00 :        7c430:       ja     7c5a6 <_int_malloc+0x8a6>
  0.00 :        7c436:       add    $0x77,%r9d
  0.00 :        7c43a:       jmpq   7be54 <_int_malloc+0x154>
  0.00 :        7c43f:       mov    0x8(%r13),%rsi
  0.00 :        7c443:       mov    %rsi,%rcx
  0.00 :        7c446:       and    $0xfffffffffffffff8,%rcx
  0.00 :        7c44a:       cmp    %rcx,%rbp
  0.00 :        7c44d:       ja     7cdb7 <_int_malloc+0x10b7>
  0.00 :        7c453:       mov    0x10(%r13),%rax
  0.00 :        7c457:       mov    %rcx,%r15
  0.00 :        7c45a:       mov    0x18(%r13),%rdx
  0.00 :        7c45e:       sub    %rbp,%r15
  0.00 :        7c461:       cmp    0x18(%rax),%r13
  0.00 :        7c465:       jne    7cd93 <_int_malloc+0x1093>
  0.00 :        7c46b:       cmp    0x10(%rdx),%r13
  0.00 :        7c46f:       jne    7cd93 <_int_malloc+0x1093>
  0.00 :        7c475:       cmp    $0x3ff,%rsi
  0.00 :        7c47c:       mov    %rdx,0x18(%rax)
  0.00 :        7c480:       mov    %rax,0x10(%rdx)
  0.00 :        7c484:       jbe    7c4be <_int_malloc+0x7be>
  0.00 :        7c486:       mov    0x20(%r13),%rdx
  0.00 :        7c48a:       test   %rdx,%rdx
  0.00 :        7c48d:       je     7c4be <_int_malloc+0x7be>
  0.00 :        7c48f:       cmp    0x28(%rdx),%r13
  0.00 :        7c493:       jne    7cf0b <_int_malloc+0x120b>
```

```
 0.00 :        7c499:       mov    0x28(%r13),%rsi
 0.00 :        7c49d:       cmp    %r13,0x20(%rsi)
 0.00 :        7c4a1:       jne    7ceec <_int_malloc+0x11ec>
 0.00 :        7c4a7:       cmpq   $0x0,0x20(%rax)
 0.00 :        7c4ac:       je     7ceca <_int_malloc+0x11ca>
 0.00 :        7c4b2:       mov    %rsi,0x28(%rdx)
 0.00 :        7c4b6:       mov    0x28(%r13),%rax
 0.00 :        7c4ba:       mov    %rdx,0x20(%rax)
 0.00 :        7c4be:       cmp    $0x1f,%r15
 0.00 :        7c4c2:       jbe    7c54d <_int_malloc+0x84d>
 0.00 :        7c4c8:       mov    0x68(%rbx),%rdx
 0.00 :        7c4cc:       lea    0x0(%r13,%rbp,1),%rax
 0.00 :        7c4d1:       cmp    0x18(%rdx),%r14
 0.00 :        7c4d5:       jne    7cd8a <_int_malloc+0x108a>
 0.00 :        7c4db:       cmp    $0x3ff,%rbp
 0.00 :        7c4e2:       mov    %r14,0x18(%rax)
 0.00 :        7c4e6:       mov    %rdx,0x10(%rax)
 0.00 :        7c4ea:       mov    %rax,0x18(%rdx)
 0.00 :        7c4ee:       mov    %rax,0x68(%rbx)
 0.00 :        7c4f2:       ja     7c4f8 <_int_malloc+0x7f8>
 0.00 :        7c4f4:       mov    %rax,0x60(%rbx)
 0.00 :        7c4f8:       cmp    $0x3ff,%r15
 0.00 :        7c4ff:       jbe    7c511 <_int_malloc+0x811>
 0.00 :        7c501:       movq   $0x0,0x20(%rax)
 0.00 :        7c509:       movq   $0x0,0x28(%rax)
 0.00 :        7c511:       lea    0x32b128(%rip),%rdx        # 3a7640 <main_arena>
 0.00 :        7c518:       mov    %r15,(%rax,%r15,1)
 0.00 :        7c51c:       cmp    %rdx,%rbx
 0.00 :        7c51f:       setne  %dl
 0.00 :        7c522:       or     $0x1,%rbp
 0.00 :        7c526:       movzbl %dl,%edx
 0.00 :        7c529:       shl    $0x2,%rdx
 0.00 :        7c52d:       or     %rdx,%rbp
 0.00 :        7c530:       mov    %r15,%rdx
 0.00 :        7c533:       or     $0x1,%rdx
 0.00 :        7c537:       mov    %rbp,0x8(%r13)
 0.00 :        7c53b:       mov    %rdx,0x8(%rax)
 0.00 :        7c53f:       jmpq   7bdc1 <_int_malloc+0xc1>
 0.00 :        7c544:       add    $0x6e,%r9d
 0.00 :        7c548:       jmpq   7be54 <_int_malloc+0x154>
 0.00 :        7c54d:       lea    0x32b0ec(%rip),%rax        # 3a7640 <main_arena>
 0.00 :        7c554:       orq    $0x1,0x8(%r13,%rcx,1)
 0.00 :        7c55a:       cmp    %rax,%rbx
 0.00 :        7c55d:       jne    7be38 <_int_malloc+0x138>
 0.00 :        7c563:       jmpq   7bdc1 <_int_malloc+0xc1>
 0.00 :        7c568:       lea    0xecae4(%rip),%rcx        # 169053 <__func__.11427>
 0.00 :        7c56f:       lea    0xec7df(%rip),%rsi        # 168d55 <__PRETTY_FUNCTION__.11753+0xb8>
 0.00 :        7c576:       lea    0xec906(%rip),%rdi        # 168e83 <__PRETTY_FUNCTION__.11753+0x1e6>
 0.00 :        7c57d:       mov    $0xd8e,%edx
 0.00 :        7c582:       callq  7a110 <__malloc_assert>
 0.00 :        7c587:       lea    0xecac5(%rip),%rcx        # 169053 <__func__.11427>
 0.00 :        7c58e:       lea    0xec7c0(%rip),%rsi        # 168d55 <__PRETTY_FUNCTION__.11753+0xb8>
 0.00 :        7c595:       lea    0xec8fe(%rip),%rdi        # 168e9a <__PRETTY_FUNCTION__.11753+0x1fd>
 0.00 :        7c59c:       mov    $0xe0f,%edx
 0.00 :        7c5a1:       callq  7a110 <__malloc_assert>
 0.00 :        7c5a6:       mov    %rbp,%rax
 0.00 :        7c5a9:       mov    $0x7e,%r9d
 0.00 :        7c5af:       shr    $0x12,%rax
 0.00 :        7c5b3:       lea    0x7c(%rax),%edx
 0.00 :        7c5b6:       cmp    $0x2,%rax
 0.00 :        7c5ba:       cmovbe %edx,%r9d
 0.00 :        7c5be:       jmpq   7be54 <_int_malloc+0x154>
 0.00 :        7c5c3:       mov    0x32abef(%rip),%edi        # 3a71b8 <check_action>
 0.00 :        7c5c9:       lea    0xec87c(%rip),%rsi        # 168e4c <__PRETTY_FUNCTION__.11753+0x1af>
```

```
 0.00 :        7c5d0:       lea    0x10(%r13),%rdx
 0.00 :        7c5d4:       mov    %rax,0x18(%rsp)
 0.00 :        7c5d9:       mov    %r8d,0x10(%rsp)
 0.00 :        7c5de:       mov    %r9d,0x8(%rsp)
 0.00 :        7c5e3:       callq  7aa50 <malloc_printerr>
 0.00 :        7c5e8:       mov    0x8(%r13),%rsi
 0.00 :        7c5ec:       mov    0x8(%rsp),%r9d
 0.00 :        7c5f1:       mov    0x10(%rsp),%r8d
 0.00 :        7c5f6:       mov    0x18(%rsp),%rax
 0.00 :        7c5fb:       jmpq   7bf60 <_int_malloc+0x260>
 0.00 :        7c600:       lea    0x32b039(%rip),%rax        # 3a7640 <main_arena>
 0.00 :        7c607:       sub    %rbp,%r10
 0.00 :        7c60a:       mov    %rdx,%r15
 0.00 :        7c60d:       lea    (%rdx,%rbp,1),%rdx
 0.00 :        7c611:       lea    0x10(%r15),%r13
 0.00 :        7c615:       cmp    %rax,%rbx
 0.00 :        7c618:       setne  %al
 0.00 :        7c61b:       or     $0x1,%rbp
 0.00 :        7c61f:       or     $0x1,%r10
 0.00 :        7c623:       movzbl %al,%eax
 0.00 :        7c626:       mov    %rdx,0x58(%rbx)
 0.00 :        7c62a:       shl    $0x2,%rax
 0.00 :        7c62e:       or     %rax,%rbp
 0.00 :        7c631:       mov    0x32d231(%rip),%eax        # 3a9868 <perturb_byte>
 0.00 :        7c637:       mov    %rbp,0x8(%r15)
 0.00 :        7c63b:       mov    %r10,0x8(%rdx)
 0.00 :        7c63f:       test   %eax,%eax
 0.00 :        7c641:       je     7bdd3 <_int_malloc+0xd3>
 0.00 :        7c647:       jmpq   7c14f <_int_malloc+0x44f>
 0.00 :        7c64c:       nopl   0x0(%rax)
 0.00 :        7c650:       mov    %rdx,%r15
 0.00 :        7c653:       mov    0x32a7ce(%rip),%rdx        # 3a6e28 <_DYNAMIC+0x2e8>
 0.00 :        7c65a:       mov    0x18(%rdx),%rdx
 0.00 :        7c65e:       mov    %rdx,0x28(%rsp)
 0.00 :        7c663:       sub    $0x1,%rdx
 0.00 :        7c667:       cmp    0x32ab02(%rip),%rbp        # 3a7170 <mp_+0x10>
 0.00 :        7c66e:       mov    %rdx,0x30(%rsp)
 0.00 :        7c673:       jb     7c800 <_int_malloc+0xb00>
 0.00 :        7c679:       mov    0x32ab0d(%rip),%ecx        # 3a718c <mp_+0x2c>
 0.00 :        7c67f:       cmp    %ecx,0x32ab03(%rip)        # 3a7188 <mp_+0x28>
 0.00 :        7c685:       jge    7c800 <_int_malloc+0xb00>
 0.00 :        7c68b:       mov    0x28(%rsp),%rax
 0.00 :        7c690:       lea    0x7(%rbp,%rax,1),%r15
 0.00 :        7c695:       mov    0x30(%rsp),%rax
 0.00 :        7c69a:       not    %rax
 0.00 :        7c69d:       and    %rax,%r15
 0.00 :        7c6a0:       cmp    %r15,%rbp
 0.00 :        7c6a3:       jb     7cf45 <_int_malloc+0x1245>
 0.00 :        7c6a9:       mov    0x58(%rbx),%r15
 0.00 :        7c6ad:       mov    $0x1,%r8d
 0.00 :        7c6b3:       mov    0x8(%r15),%rax
 0.00 :        7c6b7:       mov    %rax,%r10
 0.00 :        7c6ba:       and    $0xfffffffffffffff8,%r10
 0.00 :        7c6be:       cmp    %r14,%r15
 0.00 :        7c6c1:       lea    (%r15,%r10,1),%r13
 0.00 :        7c6c5:       je     7cf37 <_int_malloc+0x1237>
 0.00 :        7c6cb:       cmp    $0x1f,%r10
 0.00 :        7c6cf:       jbe    7cbb1 <_int_malloc+0xeb1>
 0.00 :        7c6d5:       test   $0x1,%al
 0.00 :        7c6d7:       je     7cbb1 <_int_malloc+0xeb1>
 0.00 :        7c6dd:       test   %r13,0x30(%rsp)
 0.00 :        7c6e2:       jne    7cbb1 <_int_malloc+0xeb1>
 0.00 :        7c6e8:       cmp    %r10,0x20(%rsp)
 0.00 :        7c6ed:       jbe    7cb92 <_int_malloc+0xe92>
```

```
 0.00 :        7c6f3:       lea    0x32af46(%rip),%rdx        # 3a7640 <main_arena>
 0.00 :        7c6fa:       cmp    %rdx,%rbx
 0.00 :        7c6fd:       je     7c8ef <_int_malloc+0xbef>
 0.00 :        7c703:       mov    0x20(%rsp),%rcx
 0.00 :        7c708:       mov    %r15,%r13
 0.00 :        7c70b:       and    $0xfffffffffc000000,%r13
 0.00 :        7c712:       mov    0x10(%r13),%r9
 0.00 :        7c716:       sub    %r10,%rcx
 0.00 :        7c719:       test   %rcx,%rcx
 0.00 :        7c71c:       jle    7c808 <_int_malloc+0xb08>
 0.00 :        7c722:       add    0x30(%rsp),%rcx
 0.00 :        7c727:       mov    0x30(%rsp),%rax
 0.00 :        7c72c:       not    %rax
 0.00 :        7c72f:       and    %rax,%rcx
 0.00 :        7c732:       add    %r9,%rcx
 0.00 :        7c735:       cmp    $0x4000000,%rcx
 0.00 :        7c73c:       ja     7c808 <_int_malloc+0xb08>
 0.00 :        7c742:       mov    0x18(%r13),%rdi
 0.00 :        7c746:       mov    %r15,%rdx
 0.00 :        7c749:       cmp    %rdi,%rcx
 0.00 :        7c74c:       ja     7c8a0 <_int_malloc+0xba0>
 0.00 :        7c752:       mov    %rcx,%rax
 0.00 :        7c755:       mov    %rcx,0x10(%r13)
 0.00 :        7c759:       add    %r13,%rcx
 0.00 :        7c75c:       sub    %r9,%rax
 0.00 :        7c75f:       add    %rax,0x32d10a(%rip)        # 3a9870 <arena_mem>
 0.00 :        7c766:       sub    %r15,%rcx
 0.00 :        7c769:       mov    %rax,%rsi
 0.00 :        7c76c:       add    0x878(%rbx),%rsi
 0.00 :        7c773:       or     $0x1,%rcx
 0.00 :        7c777:       mov    %rsi,0x878(%rbx)
 0.00 :        7c77e:       mov    %rcx,0x8(%r15)
 0.00 :        7c782:       cmp    0x880(%rbx),%rsi
 0.00 :        7c789:       jbe    7c792 <_int_malloc+0xa92>
 0.00 :        7c78b:       mov    %rsi,0x880(%rbx)
 0.00 :        7c792:       mov    0x8(%rdx),%rax
 0.00 :        7c796:       and    $0xfffffffffffffff8,%rax
 0.00 :        7c79a:       cmp    0x20(%rsp),%rax
 0.00 :        7c79f:       jb     7c3fe <_int_malloc+0x6fe>
 0.00 :        7c7a5:       lea    0x32ae94(%rip),%rcx        # 3a7640 <main_arena>
 0.00 :        7c7ac:       sub    %rbp,%rax
 0.00 :        7c7af:       lea    (%rdx,%rbp,1),%rsi
 0.00 :        7c7b3:       lea    0x10(%rdx),%r13
 0.00 :        7c7b7:       cmp    %rcx,%rbx
 0.00 :        7c7ba:       mov    %rsi,0x58(%rbx)
 0.00 :        7c7be:       setne  %cl
 0.00 :        7c7c1:       or     $0x1,%rbp
 0.00 :        7c7c5:       or     $0x1,%rax
 0.00 :        7c7c9:       movzbl %cl,%ecx
 0.00 :        7c7cc:       shl    $0x2,%rcx
 0.00 :        7c7d0:       or     %rcx,%rbp
 0.00 :        7c7d3:       mov    %rbp,0x8(%rdx)
 0.00 :        7c7d7:       mov    %rax,0x8(%rsi)
 0.00 :        7c7db:       test   %r13,%r13
 0.00 :        7c7de:       je     7d003 <_int_malloc+0x1303>
 0.00 :        7c7e4:       mov    0x32d07e(%rip),%eax        # 3a9868 <perturb_byte>
 0.00 :        7c7ea:       test   %eax,%eax
 0.00 :        7c7ec:       je     7bdd3 <_int_malloc+0xd3>
 0.00 :        7c7f2:       jmpq   7c14f <_int_malloc+0x44f>
 0.00 :        7c7f7:       nopw   0x0(%rax,%rax,1)
 0.00 :        7c800:       xor    %r8d,%r8d
 0.00 :        7c803:       jmpq   7c6be <_int_malloc+0x9be>
 0.00 :        7c808:       mov    0x32a959(%rip),%rsi        # 3a7168 <mp_+0x8>
 0.00 :        7c80f:       lea    0x40(%rbp),%rdi
```

```
 0.00 :        7c813:       mov    %r8b,0x10(%rsp)
 0.00 :        7c818:       mov    %r10,(%rsp)
 0.00 :        7c81c:       callq  7a190 <new_heap>
 0.00 :        7c821:       test   %rax,%rax
 0.00 :        7c824:       movzbl 0x10(%rsp),%r8d
 0.00 :        7c82a:       mov    (%rsp),%r10
 0.00 :        7c82e:       je     7cc08 <_int_malloc+0xf08>
 0.00 :        7c834:       mov    0x10(%rax),%rcx
 0.00 :        7c838:       lea    0x20(%rax),%rdx
 0.00 :        7c83c:       add    %rcx,0x32d02d(%rip)        # 3a9870 <arena_mem>
 0.00 :        7c843:       sub    $0x20,%r10
 0.00 :        7c847:       mov    %r13,0x8(%rax)
 0.00 :        7c84b:       mov    %rbx,(%rax)
 0.00 :        7c84e:       and    $0xfffffffffffffff0,%r10
 0.00 :        7c852:       mov    %rdx,0x58(%rbx)
 0.00 :        7c856:       mov    %rcx,%rsi
 0.00 :        7c859:       add    0x878(%rbx),%rsi
 0.00 :        7c860:       sub    $0x20,%rcx
 0.00 :        7c864:       or     $0x1,%rcx
 0.00 :        7c868:       cmp    $0x1f,%r10
 0.00 :        7c86c:       mov    %rsi,0x878(%rbx)
 0.00 :        7c873:       mov    %rcx,0x28(%rax)
 0.00 :        7c877:       lea    0x10(%r10),%rcx
 0.00 :        7c87b:       lea    (%r15,%rcx,1),%rax
 0.00 :        7c87f:       movq   $0x1,0x8(%rax)
 0.00 :        7c887:       ja     7cbd0 <_int_malloc+0xed0>
 0.00 :        7c88d:       mov    %rcx,%rdi
 0.00 :        7c890:       mov    %rcx,(%rax)
 0.00 :        7c893:       or     $0x1,%rdi
 0.00 :        7c897:       mov    %rdi,0x8(%r15)
 0.00 :        7c89b:       jmpq   7c782 <_int_malloc+0xa82>
 0.00 :        7c8a0:       mov    %rcx,%rsi
 0.00 :        7c8a3:       mov    $0x3,%edx
 0.00 :        7c8a8:       mov    %rcx,0x18(%rsp)
 0.00 :        7c8ad:       sub    %rdi,%rsi
 0.00 :        7c8b0:       add    %r13,%rdi
 0.00 :        7c8b3:       mov    %r8b,0x10(%rsp)
 0.00 :        7c8b8:       mov    %r9,0x8(%rsp)
 0.00 :        7c8bd:       mov    %r10,(%rsp)
 0.00 :        7c8c1:       callq  e4d30 <mprotect>
 0.00 :        7c8c6:       test   %eax,%eax
 0.00 :        7c8c8:       mov    0x18(%rsp),%rcx
 0.00 :        7c8cd:       movzbl 0x10(%rsp),%r8d
 0.00 :        7c8d3:       mov    0x8(%rsp),%r9
 0.00 :        7c8d8:       mov    (%rsp),%r10
 0.00 :        7c8dc:       jne    7c808 <_int_malloc+0xb08>
 0.00 :        7c8e2:       mov    %rcx,0x18(%r13)
 0.00 :        7c8e6:       mov    0x58(%rbx),%rdx
 0.00 :        7c8ea:       jmpq   7c752 <_int_malloc+0xa52>
 0.00 :        7c8ef:       mov    0x32a872(%rip),%rax        # 3a7168 <mp_+0x8>
 0.00 :        7c8f6:       mov    0x30(%rsp),%r14
 0.00 :        7c8fb:       lea    0x20(%rbp,%rax,1),%rdx
 0.00 :        7c900:       mov    0x32ad3e(%rip),%eax        # 3a7644 <main_arena+0x4>
 0.00 :        7c906:       mov    %rdx,%rcx
 0.00 :        7c909:       sub    %r10,%rcx
 0.00 :        7c90c:       and    $0x2,%eax
 0.00 :        7c90f:       cmove  %rcx,%rdx
 0.00 :        7c913:       mov    0x30(%rsp),%rcx
 0.00 :        7c918:       add    %rdx,%r14
 0.00 :        7c91b:       not    %rcx
 0.00 :        7c91e:       and    %rcx,%r14
 0.00 :        7c921:       mov    %rcx,0x28(%rsp)
 0.00 :        7c926:       test   %r14,%r14
 0.00 :        7c929:       mov    %r14,%rcx
```

```
 0.00 :        7c92c:       jle    7c9d1 <_int_malloc+0xcd1>
 0.00 :        7c932:       mov    0x32a437(%rip),%rax        # 3a6d70 <_DYNAMIC+0x230>
 0.00 :        7c939:       mov    %r10,(%rsp)
 0.00 :        7c93d:       mov    %r14,%rdi
 0.00 :        7c940:       mov    %r14,0x18(%rsp)
 0.00 :        7c945:       callq  *(%rax)
 0.00 :        7c947:       mov    %rax,%rdx
 0.00 :        7c94a:       mov    0x32acf4(%rip),%eax        # 3a7644 <main_arena+0x4>
 0.00 :        7c950:       mov    0x18(%rsp),%rcx
 0.00 :        7c955:       mov    (%rsp),%r10
 0.00 :        7c959:       and    $0x2,%eax
 0.00 :        7c95c:       test   %rdx,%rdx
 0.00 :        7c95f:       je     7c9d1 <_int_malloc+0xcd1>
 0.00 :        7c961:       mov    0x32a508(%rip),%rax        # 3a6e70 <_DYNAMIC+0x330>
 0.00 :        7c968:       mov    (%rax),%rax
 0.00 :        7c96b:       test   %rax,%rax
 0.00 :        7c96e:       je     7c984 <_int_malloc+0xc84>
 0.00 :        7c970:       mov    %rdx,0x10(%rsp)
 0.00 :        7c975:       mov    %r10,(%rsp)
 0.00 :        7c979:       callq  *%rax
 0.00 :        7c97b:       mov    0x10(%rsp),%rdx
 0.00 :        7c980:       mov    (%rsp),%r10
 0.00 :        7c984:       mov    $0x1,%edi
 0.00 :        7c989:       xor    %eax,%eax
 0.00 :        7c98b:       cmpq   $0x0,0x32a81d(%rip)        # 3a71b0 <mp_+0x50>
 0.00 :        7c993:       jne    7c99c <_int_malloc+0xc9c>
 0.00 :        7c995:       mov    %rdx,0x32a814(%rip)        # 3a71b0 <mp_+0x50>
 0.00 :        7c99c:       mov    %r14,%rsi
 0.00 :        7c99f:       add    0x32b512(%rip),%rsi        # 3a7eb8 <main_arena+0x878>
 0.00 :        7c9a6:       cmp    %rdx,%r13
 0.00 :        7c9a9:       mov    %rsi,%rcx
 0.00 :        7c9ac:       mov    %rsi,0x32b505(%rip)        # 3a7eb8 <main_arena+0x878>
 0.00 :        7c9b3:       jne    7ca12 <_int_malloc+0xd12>
 0.00 :        7c9b5:       test   %dil,%dil
 0.00 :        7c9b8:       je     7ca12 <_int_malloc+0xd12>
 0.00 :        7c9ba:       add    %r14,%r10
 0.00 :        7c9bd:       mov    0x32acd4(%rip),%rdx        # 3a7698 <main_arena+0x58>
 0.00 :        7c9c4:       or     $0x1,%r10
 0.00 :        7c9c8:       mov    %r10,0x8(%r15)
 0.00 :        7c9cc:       jmpq   7c782 <_int_malloc+0xa82>
 0.00 :        7c9d1:       test   %eax,%eax
 0.00 :        7c9d3:       jne    7c9e5 <_int_malloc+0xce5>
 0.00 :        7c9d5:       mov    0x30(%rsp),%rcx
 0.00 :        7c9da:       add    %r10,%rcx
 0.00 :        7c9dd:       add    %r14,%rcx
 0.00 :        7c9e0:       and    0x28(%rsp),%rcx
 0.00 :        7c9e5:       cmp    $0xfffff,%rcx
 0.00 :        7c9ec:       mov    $0x100000,%r14d
 0.00 :        7c9f2:       cmova  %rcx,%r14
 0.00 :        7c9f6:       cmp    %r14,%rbp
 0.00 :        7c9f9:       jb     7cc4f <_int_malloc+0xf4f>
 0.00 :        7c9ff:       mov    0x32b4b2(%rip),%rsi        # 3a7eb8 <main_arena+0x878>
 0.00 :        7ca06:       mov    0x32ac8b(%rip),%rdx        # 3a7698 <main_arena+0x58>
 0.00 :        7ca0d:       jmpq   7c782 <_int_malloc+0xa82>
 0.00 :        7ca12:       testb  $0x2,0x32ac2b(%rip)        # 3a7644 <main_arena+0x4>
 0.00 :        7ca19:       jne    7cc21 <_int_malloc+0xf21>
 0.00 :        7ca1f:       test   %r10,%r10
 0.00 :        7ca22:       setne  %al
 0.00 :        7ca25:       cmp    %rdx,%r13
 0.00 :        7ca28:       ja     7ccbc <_int_malloc+0xfbc>
 0.00 :        7ca2e:       test   %al,%al
 0.00 :        7ca30:       je     7ca42 <_int_malloc+0xd42>
 0.00 :        7ca32:       mov    %rdx,%rax
 0.00 :        7ca35:       sub    %r13,%rax
```

```
  0.00 :         7ca38:       add    %rax,%rsi
  0.00 :         7ca3b:       mov    %rsi,0x32b476(%rip)        # 3a7eb8 <main_arena+0x878>
  0.00 :         7ca42:       mov    %rdx,%rcx
  0.00 :         7ca45:       and    $0xf,%ecx
  0.00 :         7ca48:       je     7cb88 <_int_malloc+0xe88>
  0.00 :         7ca4e:       mov    $0x10,%eax
  0.00 :         7ca53:       sub    %rcx,%rax
  0.00 :         7ca56:       lea    (%rdx,%rax,1),%r13
  0.00 :         7ca5a:       lea    (%rax,%r10,1),%rcx
  0.00 :         7ca5e:       lea    (%rcx,%r14,1),%rax
  0.00 :         7ca62:       add    %rdx,%rax
  0.00 :         7ca65:       mov    %rcx,%rdx
  0.00 :         7ca68:       sub    %rax,%rdx
  0.00 :         7ca6b:       add    0x30(%rsp),%rax
  0.00 :         7ca70:       and    0x28(%rsp),%rax
  0.00 :         7ca75:       add    %rax,%rdx
  0.00 :         7ca78:       test   %rdx,%rdx
  0.00 :         7ca7b:       mov    %rdx,%r14
  0.00 :         7ca7e:       js     7cb69 <_int_malloc+0xe69>
  0.00 :         7ca84:       mov    0x32a2e5(%rip),%rcx        # 3a6d70 <_DYNAMIC+0x230>
  0.00 :         7ca8b:       mov    %rdx,%rdi
  0.00 :         7ca8e:       mov    %rdx,0x10(%rsp)
  0.00 :         7ca93:       mov    %r10,(%rsp)
  0.00 :         7ca97:       callq  *(%rcx)
  0.00 :         7ca99:       test   %rax,%rax
  0.00 :         7ca9c:       mov    0x10(%rsp),%rdx
  0.00 :         7caa1:       mov    (%rsp),%r10
  0.00 :         7caa5:       je     7cb47 <_int_malloc+0xe47>
  0.00 :         7caab:       mov    0x32a3be(%rip),%rcx        # 3a6e70 <_DYNAMIC+0x330>
  0.00 :         7cab2:       mov    (%rcx),%rcx
  0.00 :         7cab5:       test   %rcx,%rcx
  0.00 :         7cab8:       jne    7cb1d <_int_malloc+0xe1d>
  0.00 :         7caba:       mov    0x32b3f7(%rip),%rcx        # 3a7eb8 <main_arena+0x878>
  0.00 :         7cac1:       sub    %r13,%rax
  0.00 :         7cac4:       lea    (%rdx,%rcx,1),%rsi
  0.00 :         7cac8:       mov    %r13,0x32abc9(%rip)        # 3a7698 <main_arena+0x58>
  0.00 :         7cacf:       add    %rax,%r14
  0.00 :         7cad2:       or     $0x1,%r14
  0.00 :         7cad6:       test   %r10,%r10
  0.00 :         7cad9:       mov    %r14,0x8(%r13)
  0.00 :         7cadd:       mov    %rsi,0x32b3d4(%rip)        # 3a7eb8 <main_arena+0x878>
  0.00 :         7cae4:       je     7cb15 <_int_malloc+0xe15>
  0.00 :         7cae6:       sub    $0x20,%r10
  0.00 :         7caea:       and    $0xfffffffffffffff0,%r10
  0.00 :         7caee:       mov    %r10,%rax
  0.00 :         7caf1:       or     $0x1,%rax
  0.00 :         7caf5:       cmp    $0x1f,%r10
  0.00 :         7caf9:       mov    %rax,0x8(%r15)
  0.00 :         7cafd:       movq   $0x11,0x8(%r15,%r10,1)
  0.00 :         7cb06:       movq   $0x11,0x18(%r15,%r10,1)
  0.00 :         7cb0f:       ja     7cce8 <_int_malloc+0xfe8>
  0.00 :         7cb15:       mov    %r13,%rdx
  0.00 :         7cb18:       jmpq   7c782 <_int_malloc+0xa82>
  0.00 :         7cb1d:       mov    %rax,0x18(%rsp)
  0.00 :         7cb22:       mov    %rdx,0x10(%rsp)
  0.00 :         7cb27:       mov    %r10,(%rsp)
  0.00 :         7cb2b:       callq  *%rcx
  0.00 :         7cb2d:       mov    0x32b384(%rip),%rcx        # 3a7eb8 <main_arena+0x878>
  0.00 :         7cb34:       mov    0x18(%rsp),%rax
  0.00 :         7cb39:       mov    0x10(%rsp),%rdx
  0.00 :         7cb3e:       mov    (%rsp),%r10
  0.00 :         7cb42:       jmpq   7cac1 <_int_malloc+0xdc1>
  0.00 :         7cb47:       mov    0x32a222(%rip),%rdx        # 3a6d70 <_DYNAMIC+0x230>
  0.00 :         7cb4e:       xor    %edi,%edi
```

```
 0.00 :        7cb50:       callq  *(%rdx)
 0.00 :        7cb52:       mov    (%rsp),%r10
 0.00 :        7cb56:       xor    %edx,%edx
 0.00 :        7cb58:       xor    %r14d,%r14d
 0.00 :        7cb5b:       test   %rax,%rax
 0.00 :        7cb5e:       jne    7caba <_int_malloc+0xdba>
 0.00 :        7cb64:       jmpq   7c9ff <_int_malloc+0xcff>
 0.00 :        7cb69:       lea    0xec4ef(%rip),%rcx        # 16905f <__func__.11196>
 0.00 :        7cb70:       lea    0xec1de(%rip),%rsi        # 168d55 <__PRETTY_FUNCTION__.11753+0xb8>
 0.00 :        7cb77:       lea    0xec325(%rip),%rdi        # 168ea3 <__PRETTY_FUNCTION__.11753+0x206>
 0.00 :        7cb7e:       mov    $0xa12,%edx
 0.00 :        7cb83:       callq  7a110 <__malloc_assert>
 0.00 :        7cb88:       mov    %rdx,%r13
 0.00 :        7cb8b:       xor    %eax,%eax
 0.00 :        7cb8d:       jmpq   7ca5a <_int_malloc+0xd5a>
 0.00 :        7cb92:       lea    0xec4c6(%rip),%rcx        # 16905f <__func__.11196>
 0.00 :        7cb99:       lea    0xec1b5(%rip),%rsi        # 168d55 <__PRETTY_FUNCTION__.11753+0xb8>
 0.00 :        7cba0:       lea    0xf0921(%rip),%rdi        # 16d4c8 <__PRETTY_FUNCTION__.9328+0x20a0>
 0.00 :        7cba7:       mov    $0x944,%edx
 0.00 :        7cbac:       callq  7a110 <__malloc_assert>
 0.00 :        7cbb1:       lea    0xec4a7(%rip),%rcx        # 16905f <__func__.11196>
 0.00 :        7cbb8:       lea    0xec196(%rip),%rsi        # 168d55 <__PRETTY_FUNCTION__.11753+0xb8>
 0.00 :        7cbbf:       lea    0xf0782(%rip),%rdi        # 16d348 <__PRETTY_FUNCTION__.9328+0x1f20>
 0.00 :        7cbc6:       mov    $0x941,%edx
 0.00 :        7cbcb:       callq  7a110 <__malloc_assert>
 0.00 :        7cbd0:       movq   $0x11,0x8(%r15,%r10,1)
 0.00 :        7cbd9:       or     $0x5,%r10
 0.00 :        7cbdd:       mov    $0x1,%edx
 0.00 :        7cbe2:       mov    %r15,%rsi
 0.00 :        7cbe5:       movq   $0x10,(%rax)
 0.00 :        7cbec:       mov    %r10,0x8(%r15)
 0.00 :        7cbf0:       mov    %rbx,%rdi
 0.00 :        7cbf3:       callq  7b190 <_int_free>
 0.00 :        7cbf8:       mov    0x878(%rbx),%rsi
 0.00 :        7cbff:       mov    0x58(%rbx),%rdx
 0.00 :        7cc03:       jmpq   7c782 <_int_malloc+0xa82>
 0.00 :        7cc08:       test   %r8b,%r8b
 0.00 :        7cc0b:       je     7c68b <_int_malloc+0x98b>
 0.00 :        7cc11:       mov    0x878(%rbx),%rsi
 0.00 :        7cc18:       mov    0x58(%rbx),%rdx
 0.00 :        7cc1c:       jmpq   7c782 <_int_malloc+0xa82>
 0.00 :        7cc21:       test   $0xf,%dl
 0.00 :        7cc24:       jne    7cc9d <_int_malloc+0xf9d>
 0.00 :        7cc26:       mov    %rdx,%r13
 0.00 :        7cc29:       xor    %r14d,%r14d
 0.00 :        7cc2c:       xor    %edx,%edx
 0.00 :        7cc2e:       test   %dil,%dil
 0.00 :        7cc31:       je     7cac1 <_int_malloc+0xdc1>
 0.00 :        7cc37:       mov    0x32a132(%rip),%rax        # 3a6d70 <_DYNAMIC+0x230>
 0.00 :        7cc3e:       mov    %r10,(%rsp)
 0.00 :        7cc42:       xor    %edi,%edi
 0.00 :        7cc44:       callq  *(%rax)
 0.00 :        7cc46:       mov    (%rsp),%r10
 0.00 :        7cc4a:       jmpq   7cb56 <_int_malloc+0xe56>
 0.00 :        7cc4f:       xor    %r9d,%r9d
 0.00 :        7cc52:       mov    $0x3,%edx
 0.00 :        7cc57:       xor    %edi,%edi
 0.00 :        7cc59:       mov    $0xffffffff,%r8d
 0.00 :        7cc5f:       mov    $0x22,%ecx
 0.00 :        7cc64:       mov    %r14,%rsi
 0.00 :        7cc67:       mov    %r10,(%rsp)
 0.00 :        7cc6b:       callq  e4cd0 <mmap>
 0.00 :        7cc70:       cmp    $0xffffffffffffffff,%rax
 0.00 :        7cc74:       mov    %rax,%rdx
```

```
  0.00 :        7cc77:       mov    (%rsp),%r10
  0.00 :        7cc7b:       je     7c9ff <_int_malloc+0xcff>
  0.00 :        7cc81:       orl    $0x2,0x32a9bc(%rip)        # 3a7644 <main_arena+0x4>
  0.00 :        7cc88:       add    %r14,%rax
  0.00 :        7cc8b:       sete   %dil
  0.00 :        7cc8f:       test   %rdx,%rdx
  0.00 :        7cc92:       jne    7c98b <_int_malloc+0xc8b>
  0.00 :        7cc98:       jmpq   7c9ff <_int_malloc+0xcff>
  0.00 :        7cc9d:       lea    0xec3bb(%rip),%rcx        # 16905f <__func__.11196>
  0.00 :        7cca4:       lea    0xec0aa(%rip),%rsi        # 168d55 <__PRETTY_FUNCTION__.11753+0xb8>
  0.00 :        7ccab:       lea    0xf08fe(%rip),%rdi        # 16d5b0 <__PRETTY_FUNCTION__.9328+0x2188>
  0.00 :        7ccb2:       mov    $0xa2e,%edx
  0.00 :        7ccb7:       callq  7a110 <__malloc_assert>
  0.00 :        7ccbc:       test   %al,%al
  0.00 :        7ccbe:       je     7ca42 <_int_malloc+0xd42>
  0.00 :        7ccc4:       lea    0xf08bd(%rip),%rsi        # 16d588 <__PRETTY_FUNCTION__.9328+0x2160>
  0.00 :        7cccb:       mov    $0x3,%edi
  0.00 :        7ccd0:       callq  7aa50 <malloc_printerr>
  0.00 :        7ccd5:       mov    0x32b1dc(%rip),%rsi        # 3a7eb8 <main_arena+0x878>
  0.00 :        7ccdc:       mov    0x32a9b5(%rip),%rdx        # 3a7698 <main_arena+0x58>
  0.00 :        7cce3:       jmpq   7c782 <_int_malloc+0xa82>
  0.00 :        7cce8:       lea    0x32a951(%rip),%rdi        # 3a7640 <main_arena>
  0.00 :        7ccef:       mov    $0x1,%edx
  0.00 :        7ccf4:       mov    %r15,%rsi
  0.00 :        7ccf7:       callq  7b190 <_int_free>
  0.00 :        7ccfc:       jmpq   7c9ff <_int_malloc+0xcff>
  0.00 :        7cd01:       mov    0x68(%rbx),%rdx
  0.00 :        7cd05:       lea    0x0(%r13,%rbp,1),%rax
  0.00 :        7cd0a:       cmp    0x18(%rdx),%r14
  0.00 :        7cd0e:       jne    7cd6c <_int_malloc+0x106c>
  0.00 :        7cd10:       cmp    $0x3ff,%rcx
  0.00 :        7cd17:       mov    %r14,0x18(%rax)
  0.00 :        7cd1b:       mov    %rdx,0x10(%rax)
  0.00 :        7cd1f:       mov    %rax,0x18(%rdx)
  0.00 :        7cd23:       mov    %rax,0x68(%rbx)
  0.00 :        7cd27:       jbe    7cd39 <_int_malloc+0x1039>
  0.00 :        7cd29:       movq   $0x0,0x20(%rax)
  0.00 :        7cd31:       movq   $0x0,0x28(%rax)
  0.00 :        7cd39:       lea    0x32a900(%rip),%rdx        # 3a7640 <main_arena>
  0.00 :        7cd40:       mov    %rcx,(%rax,%rcx,1)
  0.00 :        7cd44:       cmp    %rdx,%rbx
  0.00 :        7cd47:       setne  %dl
  0.00 :        7cd4a:       or     $0x1,%rbp
  0.00 :        7cd4e:       movzbl %dl,%edx
  0.00 :        7cd51:       shl    $0x2,%rdx
  0.00 :        7cd55:       or     %rdx,%rbp
  0.00 :        7cd58:       mov    %rcx,%rdx
  0.00 :        7cd5b:       or     $0x1,%rdx
  0.00 :        7cd5f:       mov    %rbp,0x8(%r13)
  0.00 :        7cd63:       mov    %rdx,0x8(%rax)
  0.00 :        7cd67:       jmpq   7bdc1 <_int_malloc+0xc1>
  0.00 :        7cd6c:       lea    0xf0495(%rip),%rsi        # 16d208 <__PRETTY_FUNCTION__.9328+0x1de0>
  0.00 :        7cd73:       mov    0x32a43f(%rip),%edi        # 3a71b8 <check_action>
  0.00 :        7cd79:       lea    0x10(%r13),%rdx
  0.00 :        7cd7d:       xor    %r13d,%r13d
  0.00 :        7cd80:       callq  7aa50 <malloc_printerr>
  0.00 :        7cd85:       jmpq   7bdd3 <_int_malloc+0xd3>
  0.00 :        7cd8a:       lea    0xf049f(%rip),%rsi        # 16d230 <__PRETTY_FUNCTION__.9328+0x1e08>
  0.00 :        7cd91:       jmp    7cd73 <_int_malloc+0x1073>
  0.00 :        7cd93:       mov    0x32a41f(%rip),%edi        # 3a71b8 <check_action>
  0.00 :        7cd99:       lea    0xebff3(%rip),%rsi        # 168d93 <__PRETTY_FUNCTION__.11753+0xf6>
  0.00 :        7cda0:       mov    %r13,%rdx
  0.00 :        7cda3:       mov    %rcx,0x18(%rsp)
  0.00 :        7cda8:       callq  7aa50 <malloc_printerr>
```

```
 0.00 :        7cdad:       mov    0x18(%rsp),%rcx
 0.00 :        7cdb2:       jmpq   7c4be <_int_malloc+0x7be>
 0.00 :        7cdb7:       lea    0xec295(%rip),%rcx        # 169053 <__func__.11427>
 0.00 :        7cdbe:       lea    0xebf90(%rip),%rsi        # 168d55 <__PRETTY_FUNCTION__.11753+0xb8>
 0.00 :        7cdc5:       lea    0xf04ec(%rip),%rdi        # 16d2b8 <__PRETTY_FUNCTION__.9328+0x1e90>
 0.00 :        7cdcc:       mov    $0xe20,%edx
 0.00 :        7cdd1:       callq  7a110 <__malloc_assert>
 0.00 :        7cdd6:       mov    0x32a3dc(%rip),%edi        # 3a71b8 <check_action>
 0.00 :        7cddc:       lea    0xebfb0(%rip),%rsi        # 168d93 <__PRETTY_FUNCTION__.11753+0xf6>
 0.00 :        7cde3:       mov    %r13,%rdx
 0.00 :        7cde6:       mov    %rcx,0x18(%rsp)
 0.00 :        7cdeb:       callq  7aa50 <malloc_printerr>
 0.00 :        7cdf0:       mov    0x18(%rsp),%rcx
 0.00 :        7cdf5:       jmpq   7c3be <_int_malloc+0x6be>
 0.00 :        7cdfa:       cmp    %r13,0x28(%rdx)
 0.00 :        7cdfe:       jne    7ce92 <_int_malloc+0x1192>
 0.00 :        7ce04:       mov    0x28(%r13),%rsi
 0.00 :        7ce08:       cmp    0x20(%rsi),%r13
 0.00 :        7ce0c:       jne    7ce73 <_int_malloc+0x1173>
 0.00 :        7ce0e:       cmpq   $0x0,0x20(%rax)
 0.00 :        7ce13:       je     7ce51 <_int_malloc+0x1151>
 0.00 :        7ce15:       mov    %rsi,0x28(%rdx)
 0.00 :        7ce19:       mov    0x28(%r13),%rax
 0.00 :        7ce1d:       mov    %rdx,0x20(%rax)
 0.00 :        7ce21:       jmpq   7c3be <_int_malloc+0x6be>
 0.00 :        7ce26:       lea    0xf0383(%rip),%rsi        # 16d1b0 <__PRETTY_FUNCTION__.9328+0x1d88>
 0.00 :        7ce2d:       jmpq   7cd73 <_int_malloc+0x1073>
 0.00 :        7ce32:       lea    0xec21a(%rip),%rcx        # 169053 <__func__.11427>
 0.00 :        7ce39:       lea    0xebf15(%rip),%rsi        # 168d55 <__PRETTY_FUNCTION__.11753+0xb8>
 0.00 :        7ce40:       lea    0xec021(%rip),%rdi        # 168e68 <__PRETTY_FUNCTION__.11753+0x1cb>
 0.00 :        7ce47:       mov    $0xd80,%edx
 0.00 :        7ce4c:       callq  7a110 <__malloc_assert>
 0.00 :        7ce51:       cmp    %r13,%rdx
 0.00 :        7ce54:       je     7ceb1 <_int_malloc+0x11b1>
 0.00 :        7ce56:       mov    %rdx,0x20(%rax)
 0.00 :        7ce5a:       mov    0x20(%r13),%rdx
 0.00 :        7ce5e:       mov    %rsi,0x28(%rax)
 0.00 :        7ce62:       mov    %rax,0x28(%rdx)
 0.00 :        7ce66:       mov    0x28(%r13),%rdx
 0.00 :        7ce6a:       mov    %rax,0x20(%rdx)
 0.00 :        7ce6e:       jmpq   7c3be <_int_malloc+0x6be>
 0.00 :        7ce73:       lea    0xec1d9(%rip),%rcx        # 169053 <__func__.11427>
 0.00 :        7ce7a:       lea    0xebed4(%rip),%rsi        # 168d55 <__PRETTY_FUNCTION__.11753+0xb8>
 0.00 :        7ce81:       lea    0xf0400(%rip),%rdi        # 16d288 <__PRETTY_FUNCTION__.9328+0x1e60>
 0.00 :        7ce88:       mov    $0xdc3,%edx
 0.00 :        7ce8d:       callq  7a110 <__malloc_assert>
 0.00 :        7ce92:       lea    0xec1ba(%rip),%rcx        # 169053 <__func__.11427>
 0.00 :        7ce99:       lea    0xebeb5(%rip),%rsi        # 168d55 <__PRETTY_FUNCTION__.11753+0xb8>
 0.00 :        7cea0:       lea    0xf03b1(%rip),%rdi        # 16d258 <__PRETTY_FUNCTION__.9328+0x1e30>
 0.00 :        7cea7:       mov    $0xdc3,%edx
 0.00 :        7ceac:       callq  7a110 <__malloc_assert>
 0.00 :        7ceb1:       mov    %rax,0x28(%rax)
 0.00 :        7ceb5:       mov    %rax,0x20(%rax)
 0.00 :        7ceb9:       jmpq   7c3be <_int_malloc+0x6be>
 0.00 :        7cebe:       lea    0xf0313(%rip),%rsi        # 16d1d8 <__PRETTY_FUNCTION__.9328+0x1db0>
 0.00 :        7cec5:       jmpq   7cd73 <_int_malloc+0x1073>
 0.00 :        7ceca:       cmp    %r13,%rdx
 0.00 :        7cecd:       je     7cf2a <_int_malloc+0x122a>
 0.00 :        7cecf:       mov    %rdx,0x20(%rax)
 0.00 :        7ced3:       mov    0x20(%r13),%rdx
 0.00 :        7ced7:       mov    %rsi,0x28(%rax)
 0.00 :        7cedb:       mov    %rax,0x28(%rdx)
 0.00 :        7cedf:       mov    0x28(%r13),%rdx
 0.00 :        7cee3:       mov    %rax,0x20(%rdx)
```

```
    0.00 :          7cee7:       jmpq   7c4be <_int_malloc+0x7be>
    0.00 :          7ceec:       lea    0xec160(%rip),%rcx        # 169053 <__func__.11427>
    0.00 :          7cef3:       lea    0xebe5b(%rip),%rsi        # 168d55 <__PRETTY_FUNCTION__.11753+0xb8>
    0.00 :          7cefa:       lea    0xf0387(%rip),%rdi        # 16d288 <__PRETTY_FUNCTION__.9328+0x1e60>
    0.00 :          7cf01:       mov    $0xe25,%edx
    0.00 :          7cf06:       callq  7a110 <__malloc_assert>
    0.00 :          7cf0b:       lea    0xec141(%rip),%rcx        # 169053 <__func__.11427>
    0.00 :          7cf12:       lea    0xebe3c(%rip),%rsi        # 168d55 <__PRETTY_FUNCTION__.11753+0xb8>
    0.00 :          7cf19:       lea    0xf0338(%rip),%rdi        # 16d258 <__PRETTY_FUNCTION__.9328+0x1e30>
    0.00 :          7cf20:       mov    $0xe25,%edx
    0.00 :          7cf25:       callq  7a110 <__malloc_assert>
    0.00 :          7cf2a:       mov    %rax,0x28(%rax)
    0.00 :          7cf2e:       mov    %rax,0x20(%rax)
    0.00 :          7cf32:       jmpq   7c4be <_int_malloc+0x7be>
    0.00 :          7cf37:       test   %r10,%r10
    0.00 :          7cf3a:       je     7c6f3 <_int_malloc+0x9f3>
    0.00 :          7cf40:       jmpq   7c6cb <_int_malloc+0x9cb>
    0.00 :          7cf45:       xor    %r9d,%r9d
    0.00 :          7cf48:       xor    %edi,%edi
    0.00 :          7cf4a:       mov    $0xffffffff,%r8d
    0.00 :          7cf50:       mov    $0x22,%ecx
    0.00 :          7cf55:       mov    $0x3,%edx
    0.00 :          7cf5a:       mov    %r15,%rsi
    0.00 :          7cf5d:       callq  e4cd0 <mmap>
    0.00 :          7cf62:       cmp    $0xffffffffffffffff,%rax
    0.00 :          7cf66:       je     7c6a9 <_int_malloc+0x9a9>
    0.00 :          7cf6c:       lea    0x10(%rax),%r13
    0.00 :          7cf70:       test   $0xf,%r13b
    0.00 :          7cf74:       jne    7cfc5 <_int_malloc+0x12c5>
    0.00 :          7cf76:       mov    %r15,%rdx
    0.00 :          7cf79:       or     $0x2,%rdx
    0.00 :          7cf7d:       mov    %rdx,0x8(%rax)
    0.00 :          7cf81:       mov    0x32a201(%rip),%eax        # 3a7188 <mp_+0x28>
    0.00 :          7cf87:       add    $0x1,%eax
    0.00 :          7cf8a:       cmp    0x32a200(%rip),%eax        # 3a7190 <mp_+0x30>
    0.00 :          7cf90:       mov    %eax,0x32a1f2(%rip)        # 3a7188 <mp_+0x28>
    0.00 :          7cf96:       jle    7cf9e <_int_malloc+0x129e>
    0.00 :          7cf98:       mov    %eax,0x32a1f2(%rip)        # 3a7190 <mp_+0x30>
    0.00 :          7cf9e:       add    0x32a1f3(%rip),%r15        # 3a7198 <mp_+0x38>
    0.00 :          7cfa5:       cmp    0x32a1f4(%rip),%r15        # 3a71a0 <mp_+0x40>
    0.00 :          7cfac:       mov    %r15,0x32a1e5(%rip)        # 3a7198 <mp_+0x38>
    0.00 :          7cfb3:       jbe    7c7db <_int_malloc+0xadb>
    0.00 :          7cfb9:       mov    %r15,0x32a1e0(%rip)        # 3a71a0 <mp_+0x40>
    0.00 :          7cfc0:       jmpq   7c7db <_int_malloc+0xadb>
    0.00 :          7cfc5:       lea    0xec093(%rip),%rcx        # 16905f <__func__.11196>
    0.00 :          7cfcc:       lea    0xebd82(%rip),%rsi        # 168d55 <__PRETTY_FUNCTION__.11753+0xb8>
    0.00 :          7cfd3:       lea    0xf030e(%rip),%rdi        # 16d2e8 <__PRETTY_FUNCTION__.9328+0x1ec0>
    0.00 :          7cfda:       mov    $0x910,%edx
    0.00 :          7cfdf:       callq  7a110 <__malloc_assert>
    0.00 :          7cfe4:       lea    0xec068(%rip),%rcx        # 169053 <__func__.11427>
    0.00 :          7cfeb:       lea    0xebd63(%rip),%rsi        # 168d55 <__PRETTY_FUNCTION__.11753+0xb8>
    0.00 :          7cff2:       lea    0xebe8a(%rip),%rdi        # 168e83 <__PRETTY_FUNCTION__.11753+0x1e6>
    0.00 :          7cff9:       mov    $0xd8a,%edx
    0.00 :          7cffe:       callq  7a110 <__malloc_assert>
    0.00 :          7d003:       xor    %r13d,%r13d
    0.00 :          7d006:       jmpq   7bdd3 <_int_malloc+0xd3>
 Percent |       Source code & Disassembly of ld-2.17.so
-----------------------------------------------
         :
         :
         :
         :       Disassembly of section .text:
         :
         :       0000000000008d60 <do_lookup_x>:
```

```
 0.00 :        8d60:       push   %r15
 0.00 :        8d62:       mov    %rdx,%r15
 0.00 :        8d65:       push   %r14
 0.00 :        8d67:       push   %r13
 0.00 :        8d69:       mov    %rsi,%r13
 0.00 :        8d6c:       push   %r12
 0.00 :        8d6e:       push   %rbp
 0.00 :        8d6f:       mov    %r8,%rbp
 0.00 :        8d72:       push   %rbx
 0.00 :        8d73:       sub    $0xa8,%rsp
 0.00 :        8d7a:       mov    0x8(%r9),%r12d
 0.00 :        8d7e:       mov    0xe8(%rsp),%rax
 0.00 :        8d86:       mov    0xe0(%rsp),%rbx
 0.00 :        8d8e:       mov    0xf8(%rsp),%r8
 0.00 :        8d96:       mov    %rdi,0x78(%rsp)
 0.00 :        8d9b:       mov    %rcx,0x88(%rsp)
 0.00 :        8da3:       mov    %rax,0x68(%rsp)
 0.00 :        8da8:       mov    0xf0(%rsp),%eax
 0.00 :        8daf:       mov    %eax,0x94(%rsp)
 0.00 :        8db6:       mov    0x100(%rsp),%eax
 0.00 :        8dbd:       mov    %eax,0x9c(%rsp)
 0.00 :        8dc4:       mov    %esi,%eax
 0.00 :        8dc6:       mov    (%r9),%r14
 0.00 :        8dc9:       mov    %rsi,%r9
 0.00 :        8dcc:       and    $0x3f,%eax
 0.00 :        8dcf:       shr    $0x6,%r9
 0.00 :        8dd3:       mov    %eax,%r11d
 0.00 :        8dd6:       mov    (%r14,%rbx,8),%rax
 0.00 :        8dda:       movl   $0x0,0x90(%rsp)
 0.00 :        8de5:       movq   $0x0,0x60(%rsp)
 0.00 :        8dee:       mov    0x28(%rax),%rdi
 0.00 :        8df2:       cmp    %r8,%rdi
 0.00 :        8df5:       mov    %rdi,0x70(%rsp)
 0.00 :        8dfa:       je     8f10 <do_lookup_x+0x1b0>
 0.00 :        8e00:       testb  $0x2,0x9c(%rsp)
 0.00 :        8e08:       je     8e17 <do_lookup_x+0xb7>
 0.00 :        8e0a:       testb  $0x3,0x314(%rdi)
 0.00 :        8e11:       je     8f10 <do_lookup_x+0x1b0>
 0.00 :        8e17:       testb  $0x20,0x315(%rdi)
 0.00 :        8e1e:       jne    8f10 <do_lookup_x+0x1b0>
 0.00 :        8e24:       testb  $0x8,0x218d75(%rip)        # 221ba0 <_rtld_global_ro>
 0.00 :        8e2b:       jne    93d0 <do_lookup_x+0x670>
 0.00 :        8e31:       mov    0x2ec(%rdi),%eax
 0.00 :        8e37:       test   %eax,%eax
 0.00 :        8e39:       je     8f10 <do_lookup_x+0x1b0>
 0.00 :        8e3f:       mov    0x70(%rdi),%rdx
 0.00 :        8e43:       mov    0x8(%rdx),%rdx
 0.00 :        8e47:       mov    %rdx,0x38(%rsp)
 0.00 :        8e4c:       mov    0x68(%rdi),%rdx
 0.00 :        8e50:       mov    0x8(%rdx),%rdx
50.00 :        8e54:       mov    %rdx,0x80(%rsp)
 0.00 :        8e5c:       mov    0x2f8(%rdi),%rdx
 0.00 :        8e63:       test   %rdx,%rdx
 0.00 :        8e66:       je     9529 <do_lookup_x+0x7c9>
 0.00 :        8e6c:       mov    0x2f0(%rdi),%ecx
 0.00 :        8e72:       mov    %r13,%rsi
 0.00 :        8e75:       and    %r9d,%ecx
 0.00 :        8e78:       mov    (%rdx,%rcx,8),%r10
 0.00 :        8e7c:       mov    %r11d,%ecx
 0.00 :        8e7f:       mov    %r10,%rdx
 0.00 :        8e82:       shr    %cl,%rdx
 0.00 :        8e85:       movzbl 0x2f4(%rdi),%ecx
 0.00 :        8e8c:       shr    %cl,%rsi
 0.00 :        8e8f:       mov    %rsi,0x30(%rsp)
```

```
 0.00 :        8e94:       movzbl 0x30(%rsp),%ecx
 0.00 :        8e99:       and    $0x3f,%ecx
 0.00 :        8e9c:       shr    %cl,%r10
 0.00 :        8e9f:       and    %r10,%rdx
 0.00 :        8ea2:       and    $0x1,%edx
 0.00 :        8ea5:       jne    9470 <do_lookup_x+0x710>
 0.00 :        8eab:       movl   $0x0,0x98(%rsp)
 0.00 :        8eb6:       cmpl   $0x1,0x90(%rsp)
 0.00 :        8ebe:       je     8f38 <do_lookup_x+0x1d8>
 0.00 :        8ec0:       mov    0x98(%rsp),%edx
 0.00 :        8ec7:       test   %edx,%edx
 0.00 :        8ec9:       jne    8f10 <do_lookup_x+0x1b0>
 0.00 :        8ecb:       mov    0x68(%rsp),%rax
 0.00 :        8ed0:       test   %rax,%rax
 0.00 :        8ed3:       je     8f10 <do_lookup_x+0x1b0>
 0.00 :        8ed5:       mov    0x10(%rax),%rdi
 0.00 :        8ed9:       test   %rdi,%rdi
 0.00 :        8edc:       je     8f10 <do_lookup_x+0x1b0>
 0.00 :        8ede:       mov    0x70(%rsp),%rsi
 0.00 :        8ee3:       mov    %r8,(%rsp)
 0.00 :        8ee7:       mov    %r9,0x10(%rsp)
 0.00 :        8eec:       mov    %r11b,0x8(%rsp)
 0.00 :        8ef1:       callq  fa00 <_dl_name_match_p>
 0.00 :        8ef6:       test   %eax,%eax
 0.00 :        8ef8:       mov    (%rsp),%r8
 0.00 :        8efc:       mov    0x10(%rsp),%r9
 0.00 :        8f01:       movzbl 0x8(%rsp),%r11d
 0.00 :        8f07:       jne    9468 <do_lookup_x+0x708>
 0.00 :        8f0d:       nopl   (%rax)
 0.00 :        8f10:       add    $0x1,%rbx
50.00 :        8f14:       cmp    %rbx,%r12
 0.00 :        8f17:       ja     8dd6 <do_lookup_x+0x76>
 0.00 :        8f1d:       xor    %eax,%eax
 0.00 :        8f1f:       add    $0xa8,%rsp
 0.00 :        8f26:       pop    %rbx
 0.00 :        8f27:       pop    %rbp
 0.00 :        8f28:       pop    %r12
 0.00 :        8f2a:       pop    %r13
 0.00 :        8f2c:       pop    %r14
 0.00 :        8f2e:       pop    %r15
 0.00 :        8f30:       retq
 0.00 :        8f31:       nopl   0x0(%rax)
 0.00 :        8f38:       mov    0x60(%rsp),%rax
 0.00 :        8f3d:       test   %rax,%rax
 0.00 :        8f40:       je     8ec0 <do_lookup_x+0x160>
 0.00 :        8f46:       movzbl 0x4(%rax),%edx
 0.00 :        8f4a:       shr    $0x4,%dl
 0.00 :        8f4d:       cmp    $0x2,%dl
 0.00 :        8f50:       je     8f80 <do_lookup_x+0x220>
 0.00 :        8f52:       cmp    $0xa,%dl
 0.00 :        8f55:       je     8fb0 <do_lookup_x+0x250>
 0.00 :        8f57:       cmp    $0x1,%dl
 0.00 :        8f5a:       jne    8ec0 <do_lookup_x+0x160>
 0.00 :        8f60:       mov    %rax,0x48(%rsp)
 0.00 :        8f65:       mov    0x48(%rsp),%rax
 0.00 :        8f6a:       mov    %rax,0x0(%rbp)
 0.00 :        8f6e:       mov    0x70(%rsp),%rax
 0.00 :        8f73:       mov    %rax,0x8(%rbp)
 0.00 :        8f77:       mov    $0x1,%eax
 0.00 :        8f7c:       jmp    8f1f <do_lookup_x+0x1bf>
 0.00 :        8f7e:       xchg   %ax,%ax
 0.00 :        8f80:       mov    0x218c66(%rip),%ecx        # 221bec <_rtld_global_ro+0x4c>
 0.00 :        8f86:       test   %ecx,%ecx
 0.00 :        8f88:       je     8f60 <do_lookup_x+0x200>
```

```
 0.00 :        8f8a:       cmpq   $0x0,0x0(%rbp)
 0.00 :        8f8f:       jne    8ec0 <do_lookup_x+0x160>
 0.00 :        8f95:       mov    %rax,0x0(%rbp)
 0.00 :        8f99:       mov    0x70(%rsp),%rax
 0.00 :        8f9e:       mov    %rax,0x8(%rbp)
 0.00 :        8fa2:       jmpq   8ec0 <do_lookup_x+0x160>
 0.00 :        8fa7:       nopw   0x0(%rax,%rax,1)
 0.00 :        8fb0:       mov    %rax,0x48(%rsp)
 0.00 :        8fb5:       mov    0x70(%rsp),%rax
 0.00 :        8fba:       lea    0x21903f(%rip),%rdx        # 222000 <_rtld_global>
 0.00 :        8fc1:       mov    0x30(%rax),%rax
 0.00 :        8fc5:       lea    (%rax,%rax,8),%rax
 0.00 :        8fc9:       shl    $0x4,%rax
 0.00 :        8fcd:       lea    0x20(%rdx,%rax,1),%rdx
 0.00 :        8fd2:       mov    %rdx,0x38(%rsp)
 0.00 :        8fd7:       mov    %rdx,%rdi
 0.00 :        8fda:       callq  *0x219f20(%rip)        # 222f00 <_rtld_global+0xf00>
 0.00 :        8fe0:       mov    0x38(%rsp),%rdx
 0.00 :        8fe5:       mov    0x38(%rsp),%rcx
 0.00 :        8fea:       mov    0x28(%rdx),%rdx
 0.00 :        8fee:       mov    0x30(%rcx),%r15
 0.00 :        8ff2:       test   %rdx,%rdx
 0.00 :        8ff5:       mov    %rdx,0x40(%rsp)
 0.00 :        8ffa:       je     91ca <do_lookup_x+0x46a>
 0.00 :        9000:       mov    %r13,%rax
 0.00 :        9003:       xor    %edx,%edx
 0.00 :        9005:       lea    -0x2(%r15),%rcx
 0.00 :        9009:       div    %r15
 0.00 :        900c:       mov    %r13,%rax
 0.00 :        900f:       mov    %r13,%r8
 0.00 :        9012:       mov    %rbp,0x58(%rsp)
 0.00 :        9017:       mov    %rcx,0x50(%rsp)
 0.00 :        901c:       mov    %rdx,%r14
 0.00 :        901f:       xor    %edx,%edx
 0.00 :        9021:       div    %rcx
 0.00 :        9024:       lea    0x1(%rdx),%r12
 0.00 :        9028:       mov    0x78(%rsp),%rdx
 0.00 :        902d:       mov    %r12,%rbx
 0.00 :        9030:       mov    %r12,%rbp
 0.00 :        9033:       shl    $0x5,%rbx
 0.00 :        9037:       mov    %rdx,0x30(%rsp)
 0.00 :        903c:       mov    %rbx,%r13
 0.00 :        903f:       lea    0x0(%rbp,%r14,1),%rbx
 0.00 :        9044:       shl    $0x5,%r14
 0.00 :        9048:       add    0x40(%rsp),%r14
 0.00 :        904d:       jmp    9068 <do_lookup_x+0x308>
 0.00 :        904f:       nop
 0.00 :        9050:       test   %r12,%r12
 0.00 :        9053:       je     90e0 <do_lookup_x+0x380>
 0.00 :        9059:       add    %r13,%r14
 0.00 :        905c:       cmp    %rbx,%r15
 0.00 :        905f:       lea    (%rbx,%rbp,1),%rax
 0.00 :        9063:       jbe    90d0 <do_lookup_x+0x370>
 0.00 :        9065:       mov    %rax,%rbx
 0.00 :        9068:       mov    (%r14),%eax
 0.00 :        906b:       mov    0x8(%r14),%r12
 0.00 :        906f:       cmp    %r8,%rax
 0.00 :        9072:       jne    9050 <do_lookup_x+0x2f0>
 0.00 :        9074:       mov    0x30(%rsp),%rsi
 0.00 :        9079:       mov    %r12,%rdi
 0.00 :        907c:       mov    %r8,(%rsp)
 0.00 :        9080:       callq  17700 <strcmp>
 0.00 :        9085:       test   %eax,%eax
 0.00 :        9087:       mov    (%rsp),%r8
```

```
 0.00 :        908b:       jne    9050 <do_lookup_x+0x2f0>
 0.00 :        908d:       testb  $0x2,0x9c(%rsp)
 0.00 :        9095:       mov    0x58(%rsp),%rbp
 0.00 :        909a:       je     93bb <do_lookup_x+0x65b>
 0.00 :        90a0:       mov    0x48(%rsp),%rax
 0.00 :        90a5:       mov    %rax,0x0(%rbp)
 0.00 :        90a9:       mov    0x70(%rsp),%rax
 0.00 :        90ae:       mov    %rax,0x8(%rbp)
 0.00 :        90b2:       mov    0x38(%rsp),%rdi
 0.00 :        90b7:       callq  *0x219e4b(%rip)        # 222f08 <_rtld_global+0xf08>
 0.00 :        90bd:       mov    $0x1,%eax
 0.00 :        90c2:       jmpq   8f1f <do_lookup_x+0x1bf>
 0.00 :        90c7:       nopw   0x0(%rax,%rax,1)
 0.00 :        90d0:       mov    %rbx,%r14
 0.00 :        90d3:       sub    %r15,%r14
 0.00 :        90d6:       jmpq   903f <do_lookup_x+0x2df>
 0.00 :        90db:       nopl   0x0(%rax,%rax,1)
 0.00 :        90e0:       mov    0x38(%rsp),%rcx
 0.00 :        90e5:       lea    (%r15,%r15,2),%rdx
 0.00 :        90e9:       mov    %r8,%r13
 0.00 :        90ec:       mov    0x58(%rsp),%rbp
 0.00 :        90f1:       mov    0x38(%rcx),%rcx
 0.00 :        90f5:       mov    %rcx,%rax
 0.00 :        90f8:       shl    $0x2,%rax
 0.00 :        90fc:       cmp    %rax,%rdx
 0.00 :        90ff:       ja     9230 <do_lookup_x+0x4d0>
 0.00 :        9105:       mov    %fs:0x4c,%eax
 0.00 :        910d:       test   %eax,%eax
 0.00 :        910f:       jne    9449 <do_lookup_x+0x6e9>
 0.00 :        9115:       lea    0x1(%r15),%rdi
 0.00 :        9119:       callq  fa70 <_dl_higher_prime_number>
 0.00 :        911e:       mov    $0x20,%edi
 0.00 :        9123:       mov    %rax,%rsi
 0.00 :        9126:       mov    %rax,%rbx
 0.00 :        9129:       callq  b70 <calloc@plt>
 0.00 :        912e:       test   %rax,%rax
 0.00 :        9131:       mov    %rax,%r14
 0.00 :        9134:       je     9421 <do_lookup_x+0x6c1>
 0.00 :        913a:       lea    -0x2(%rbx),%rcx
 0.00 :        913e:       test   %r15,%r15
 0.00 :        9141:       mov    %rcx,0x50(%rsp)
 0.00 :        9146:       je     9389 <do_lookup_x+0x629>
 0.00 :        914c:       mov    0x40(%rsp),%rsi
 0.00 :        9151:       shl    $0x5,%r15
 0.00 :        9155:       mov    %r13,0x30(%rsp)
 0.00 :        915a:       mov    %rcx,%r10
 0.00 :        915d:       add    %rsi,%r15
 0.00 :        9160:       mov    %r15,%r13
 0.00 :        9163:       nopl   0x0(%rax,%rax,1)
 0.00 :        9168:       mov    0x8(%rsi),%r8
 0.00 :        916c:       test   %r8,%r8
 0.00 :        916f:       je     9377 <do_lookup_x+0x617>
 0.00 :        9175:       mov    (%rsi),%r9d
 0.00 :        9178:       xor    %edx,%edx
 0.00 :        917a:       mov    0x18(%rsi),%r11
 0.00 :        917e:       mov    0x10(%rsi),%r12
 0.00 :        9182:       mov    %r9d,%edi
 0.00 :        9185:       mov    %rdi,%rax
 0.00 :        9188:       div    %rbx
 0.00 :        918b:       mov    %rdi,%rax
 0.00 :        918e:       mov    %rdx,%rcx
 0.00 :        9191:       xor    %edx,%edx
 0.00 :        9193:       div    %r10
 0.00 :        9196:       add    $0x1,%rdx
```

```
 0.00 :        919a:       mov    %rdx,%r15
 0.00 :        919d:       shl    $0x5,%r15
 0.00 :        91a1:       mov    %rcx,%rax
 0.00 :        91a4:       shl    $0x5,%rax
 0.00 :        91a8:       lea    0x8(%r14,%rax,1),%rax
 0.00 :        91ad:       nopl   (%rax)
 0.00 :        91b0:       cmpq   $0x0,(%rax)
 0.00 :        91b4:       je     9368 <do_lookup_x+0x608>
 0.00 :        91ba:       add    %rdx,%rcx
 0.00 :        91bd:       add    %r15,%rax
 0.00 :        91c0:       cmp    %rcx,%rbx
 0.00 :        91c3:       ja     91b0 <do_lookup_x+0x450>
 0.00 :        91c5:       sub    %rbx,%rcx
 0.00 :        91c8:       jmp    91a1 <do_lookup_x+0x441>
 0.00 :        91ca:       mov    %fs:0x4c,%eax
 0.00 :        91d2:       test   %eax,%eax
 0.00 :        91d4:       jne    967d <do_lookup_x+0x91d>
 0.00 :        91da:       test   %r15,%r15
 0.00 :        91dd:       nopl   (%rax)
 0.00 :        91e0:       jne    9664 <do_lookup_x+0x904>
 0.00 :        91e6:       mov    $0x1f,%esi
 0.00 :        91eb:       mov    $0x20,%edi
 0.00 :        91f0:       callq  b70 <calloc@plt>
 0.00 :        91f5:       test   %rax,%rax
 0.00 :        91f8:       mov    %rax,0x40(%rsp)
 0.00 :        91fd:       je     9421 <do_lookup_x+0x6c1>
 0.00 :        9203:       mov    0x38(%rsp),%rdx
 0.00 :        9208:       movq   $0x1d,0x50(%rsp)
 0.00 :        9211:       mov    $0x1f,%r15d
 0.00 :        9217:       mov    %rax,0x28(%rdx)
 0.00 :        921b:       mov    0x218dce(%rip),%rax        # 221ff0 <_GLOBAL_OFFSET_TABLE_+0x58>
 0.00 :        9222:       movq   $0x1f,0x30(%rdx)
 0.00 :        922a:       mov    %rax,0x40(%rdx)
 0.00 :        922e:       xchg   %ax,%ax
 0.00 :        9230:       mov    0x48(%rsp),%rax
 0.00 :        9235:       mov    %r13d,%r8d
 0.00 :        9238:       mov    (%rax),%esi
 0.00 :        923a:       add    0x80(%rsp),%rsi
 0.00 :        9242:       testb  $0x2,0x9c(%rsp)
 0.00 :        924a:       je     92d9 <do_lookup_x+0x579>
 0.00 :        9250:       mov    %r13d,%r13d
 0.00 :        9253:       xor    %edx,%edx
 0.00 :        9255:       mov    0x88(%rsp),%rdi
 0.00 :        925d:       mov    %r13,%rax
 0.00 :        9260:       mov    0x40(%rsp),%r11
 0.00 :        9265:       div    %r15
 0.00 :        9268:       mov    %r13,%rax
 0.00 :        926b:       mov    %rdx,%rcx
 0.00 :        926e:       xor    %edx,%edx
 0.00 :        9270:       divq   0x50(%rsp)
 0.00 :        9275:       lea    0x1(%rdx),%r9
 0.00 :        9279:       mov    %r9,%r10
 0.00 :        927c:       shl    $0x5,%r10
 0.00 :        9280:       mov    %rcx,%rax
 0.00 :        9283:       shl    $0x5,%rax
 0.00 :        9287:       lea    0x8(%r11,%rax,1),%rax
 0.00 :        928c:       nopl   0x0(%rax)
 0.00 :        9290:       cmpq   $0x0,(%rax)
 0.00 :        9294:       lea    -0x8(%rax),%rdx
 0.00 :        9298:       je     92aa <do_lookup_x+0x54a>
 0.00 :        929a:       add    %r9,%rcx
 0.00 :        929d:       add    %r10,%rax
 0.00 :        92a0:       cmp    %rcx,%r15
 0.00 :        92a3:       ja     9290 <do_lookup_x+0x530>
```

```
 0.00 :        92a5:       sub    %r15,%rcx
 0.00 :        92a8:       jmp    9280 <do_lookup_x+0x520>
 0.00 :        92aa:       mov    %r8d,-0x8(%rax)
 0.00 :        92ae:       mov    %rsi,(%rax)
 0.00 :        92b1:       mov    %rdi,0x8(%rax)
 0.00 :        92b5:       mov    0x108(%rsp),%rax
 0.00 :        92bd:       mov    %rax,0x18(%rdx)
 0.00 :        92c1:       mov    0x38(%rsp),%rdx
 0.00 :        92c6:       addq   $0x1,0x38(%rdx)
 0.00 :        92cb:       mov    %rdx,%rdi
 0.00 :        92ce:       callq  *0x219c34(%rip)        # 222f08 <_rtld_global+0xf08>
 0.00 :        92d4:       jmpq   8f65 <do_lookup_x+0x205>
 0.00 :        92d9:       mov    %r13d,%r13d
 0.00 :        92dc:       xor    %edx,%edx
 0.00 :        92de:       mov    0x70(%rsp),%rdi
 0.00 :        92e3:       mov    %r13,%rax
 0.00 :        92e6:       mov    0x40(%rsp),%r11
 0.00 :        92eb:       div    %r15
 0.00 :        92ee:       mov    %r13,%rax
 0.00 :        92f1:       mov    %rdx,%rcx
 0.00 :        92f4:       xor    %edx,%edx
 0.00 :        92f6:       divq   0x50(%rsp)
 0.00 :        92fb:       lea    0x1(%rdx),%r9
 0.00 :        92ff:       mov    %r9,%r10
 0.00 :        9302:       shl    $0x5,%r10
 0.00 :        9306:       mov    %rcx,%rax
 0.00 :        9309:       shl    $0x5,%rax
 0.00 :        930d:       lea    0x8(%r11,%rax,1),%rax
 0.00 :        9312:       nopw   0x0(%rax,%rax,1)
 0.00 :        9318:       cmpq   $0x0,(%rax)
 0.00 :        931c:       lea    -0x8(%rax),%rdx
 0.00 :        9320:       je     9332 <do_lookup_x+0x5d2>
 0.00 :        9322:       add    %r9,%rcx
 0.00 :        9325:       add    %r10,%rax
 0.00 :        9328:       cmp    %rcx,%r15
 0.00 :        932b:       ja     9318 <do_lookup_x+0x5b8>
 0.00 :        932d:       sub    %r15,%rcx
 0.00 :        9330:       jmp    9306 <do_lookup_x+0x5a6>
 0.00 :        9332:       mov    %r8d,-0x8(%rax)
 0.00 :        9336:       mov    %rsi,(%rax)
 0.00 :        9339:       mov    0x48(%rsp),%rax
 0.00 :        933e:       mov    %rdi,0x18(%rdx)
 0.00 :        9342:       mov    %rax,0x10(%rdx)
 0.00 :        9346:       movzbl 0x314(%rdi),%eax
 0.00 :        934d:       and    $0x3,%eax
 0.00 :        9350:       cmp    $0x2,%al
 0.00 :        9352:       jne    92c1 <do_lookup_x+0x561>
 0.00 :        9358:       orl    $0x8,0x3d4(%rdi)
 0.00 :        935f:       jmpq   92c1 <do_lookup_x+0x561>
 0.00 :        9364:       nopl   0x0(%rax)
 0.00 :        9368:       mov    %r9d,-0x8(%rax)
 0.00 :        936c:       mov    %r8,(%rax)
 0.00 :        936f:       mov    %r12,0x8(%rax)
 0.00 :        9373:       mov    %r11,0x10(%rax)
 0.00 :        9377:       add    $0x20,%rsi
 0.00 :        937b:       cmp    %r13,%rsi
 0.00 :        937e:       jne    9168 <do_lookup_x+0x408>
 0.00 :        9384:       mov    0x30(%rsp),%r13
 0.00 :        9389:       mov    0x38(%rsp),%rcx
 0.00 :        938e:       mov    0x40(%rsp),%rdi
 0.00 :        9393:       mov    %rbx,%r15
 0.00 :        9396:       callq  *0x40(%rcx)
 0.00 :        9399:       mov    0x38(%rsp),%rsi
 0.00 :        939e:       mov    0x218c4b(%rip),%rax        # 221ff0 <_GLOBAL_OFFSET_TABLE_+0x58>
```

```
 0.00 :        93a5:       mov    %r14,0x40(%rsp)
 0.00 :        93aa:       mov    %rbx,0x30(%rsi)
 0.00 :        93ae:       mov    %r14,0x28(%rsi)
 0.00 :        93b2:       mov    %rax,0x40(%rsi)
 0.00 :        93b6:       jmpq   9230 <do_lookup_x+0x4d0>
 0.00 :        93bb:       mov    0x10(%r14),%rax
 0.00 :        93bf:       mov    %rax,0x0(%rbp)
 0.00 :        93c3:       mov    0x18(%r14),%rax
 0.00 :        93c7:       mov    %rax,0x8(%rbp)
 0.00 :        93cb:       jmpq   90b2 <do_lookup_x+0x352>
 0.00 :        93d0:       mov    0x8(%rdi),%rdx
 0.00 :        93d4:       mov    0x30(%rdi),%rcx
 0.00 :        93d8:       cmpb   $0x0,(%rdx)
 0.00 :        93db:       jne    93e7 <do_lookup_x+0x687>
 0.00 :        93dd:       mov    0x2188f4(%rip),%rax        # 221cd8 <_dl_argv>
 0.00 :        93e4:       mov    (%rax),%rdx
 0.00 :        93e7:       mov    0x78(%rsp),%rsi
 0.00 :        93ec:       lea    0x13aad(%rip),%rdi        # 1cea0 <__PRETTY_FUNCTION__.4816+0x1337>
 0.00 :        93f3:       xor    %eax,%eax
 0.00 :        93f5:       mov    %r8,(%rsp)
 0.00 :        93f9:       mov    %r9,0x10(%rsp)
 0.00 :        93fe:       mov    %r11b,0x8(%rsp)
 0.00 :        9403:       callq  f8f0 <_dl_debug_printf>
 0.00 :        9408:       mov    0x70(%rsp),%rdi
 0.00 :        940d:       movzbl 0x8(%rsp),%r11d
 0.00 :        9413:       mov    0x10(%rsp),%r9
 0.00 :        9418:       mov    (%rsp),%r8
 0.00 :        941c:       jmpq   8e31 <do_lookup_x+0xd1>
 0.00 :        9421:       mov    0x38(%rsp),%rdi
 0.00 :        9426:       callq  *0x219adc(%rip)        # 222f08 <_rtld_global+0xf08>
 0.00 :        942c:       lea    0x11a3e(%rip),%rsi        # 1ae71 <intel_02_known+0x451>
 0.00 :        9433:       mov    $0x2,%edi
 0.00 :        9438:       xor    %eax,%eax
 0.00 :        943a:       callq  f9b0 <_dl_dprintf>
 0.00 :        943f:       mov    $0x7f,%edi
 0.00 :        9444:       callq  17330 <_Exit>
 0.00 :        9449:       lea    0x11f71(%rip),%rcx        # 1b3c1 <__PRETTY_FUNCTION__.9401>
 0.00 :        9450:       lea    0x11e86(%rip),%rsi        # 1b2dd <__PRETTY_FUNCTION__.9569+0xf>
 0.00 :        9457:       lea    0x13a6a(%rip),%rdi        # 1cec8 <__PRETTY_FUNCTION__.4816+0x135f>
 0.00 :        945e:       mov    $0x17a,%edx
 0.00 :        9463:       callq  15980 <__GI___assert_fail>
 0.00 :        9468:       or     $0xffffffff,%eax
 0.00 :        946b:       jmpq   8f1f <do_lookup_x+0x1bf>
 0.00 :        9470:       mov    %eax,%ecx
 0.00 :        9472:       xor    %edx,%edx
 0.00 :        9474:       mov    %r13,%rax
 0.00 :        9477:       div    %rcx
 0.00 :        947a:       mov    0x300(%rdi),%rax
 0.00 :        9481:       mov    (%rax,%rdx,4),%eax
 0.00 :        9484:       test   %eax,%eax
 0.00 :        9486:       je     8eab <do_lookup_x+0x14b>
 0.00 :        948c:       mov    0x308(%rdi),%rdx
 0.00 :        9493:       mov    0x38(%rsp),%rsi
 0.00 :        9498:       lea    (%rdx,%rax,4),%rdx
 0.00 :        949c:       jmp    94ac <do_lookup_x+0x74c>
 0.00 :        949e:       xchg   %ax,%ax
 0.00 :        94a0:       add    $0x4,%rdx
 0.00 :        94a4:       test   $0x1,%al
 0.00 :        94a6:       jne    8eab <do_lookup_x+0x14b>
 0.00 :        94ac:       mov    (%rdx),%eax
 0.00 :        94ae:       mov    %eax,%ecx
 0.00 :        94b0:       xor    %r13,%rcx
 0.00 :        94b3:       shr    %rcx
 0.00 :        94b6:       jne    94a0 <do_lookup_x+0x740>
```

```
 0.00 :        94b8:       mov    0x70(%rsp),%rax
 0.00 :        94bd:       mov    %rdx,%rcx
 0.00 :        94c0:       lea    0x60(%rsp),%r10
 0.00 :        94c5:       sub    0x308(%rax),%rcx
 0.00 :        94cc:       mov    %rdx,0x20(%rsp)
 0.00 :        94d1:       mov    %rsi,0x18(%rsp)
 0.00 :        94d6:       mov    %r8,(%rsp)
 0.00 :        94da:       mov    %r9,0x10(%rsp)
 0.00 :        94df:       mov    %r11b,0x8(%rsp)
 0.00 :        94e4:       mov    %rcx,%rax
 0.00 :        94e7:       sar    $0x2,%rax
 0.00 :        94eb:       mov    %eax,0x98(%rsp)
 0.00 :        94f2:       mov    %eax,%eax
 0.00 :        94f4:       imul   $0x18,%rax,%rdi
 0.00 :        94f8:       add    %rsi,%rdi
 0.00 :        94fb:       callq  8bd0 <check_match.9326>
 0.00 :        9500:       test   %rax,%rax
 0.00 :        9503:       mov    0x20(%rsp),%rdx
 0.00 :        9508:       mov    0x18(%rsp),%rsi
 0.00 :        950d:       mov    (%rsp),%r8
 0.00 :        9511:       mov    0x10(%rsp),%r9
 0.00 :        9516:       movzbl 0x8(%rsp),%r11d
 0.00 :        951c:       jne    8f46 <do_lookup_x+0x1e6>
 0.00 :        9522:       mov    (%rdx),%eax
 0.00 :        9524:       jmpq   94a0 <do_lookup_x+0x740>
 0.00 :        9529:       mov    $0xffffffff,%esi
 0.00 :        952e:       cmp    %rsi,(%r15)
 0.00 :        9531:       je     95d0 <do_lookup_x+0x870>
 0.00 :        9537:       mov    %eax,%ecx
 0.00 :        9539:       mov    (%r15),%rax
 0.00 :        953c:       xor    %edx,%edx
 0.00 :        953e:       div    %rcx
 0.00 :        9541:       mov    0x308(%rdi),%rax
 0.00 :        9548:       mov    (%rax,%rdx,4),%eax
 0.00 :        954b:       test   %eax,%eax
 0.00 :        954d:       mov    %eax,0x98(%rsp)
 0.00 :        9554:       je     8eb6 <do_lookup_x+0x156>
 0.00 :        955a:       mov    0x38(%rsp),%rsi
 0.00 :        955f:       jmp    958d <do_lookup_x+0x82d>
 0.00 :        9561:       nopl   0x0(%rax)
 0.00 :        9568:       mov    0x70(%rsp),%rax
 0.00 :        956d:       mov    0x98(%rsp),%edx
 0.00 :        9574:       mov    0x300(%rax),%rax
 0.00 :        957b:       mov    (%rax,%rdx,4),%eax
 0.00 :        957e:       test   %eax,%eax
 0.00 :        9580:       mov    %eax,0x98(%rsp)
 0.00 :        9587:       je     8eb6 <do_lookup_x+0x156>
 0.00 :        958d:       lea    (%rax,%rax,2),%rax
 0.00 :        9591:       lea    0x60(%rsp),%r10
 0.00 :        9596:       mov    %rsi,0x18(%rsp)
 0.00 :        959b:       mov    %r8,(%rsp)
 0.00 :        959f:       mov    %r9,0x10(%rsp)
 0.00 :        95a4:       lea    (%rsi,%rax,8),%rdi
 0.00 :        95a8:       mov    %r11b,0x8(%rsp)
 0.00 :        95ad:       callq  8bd0 <check_match.9326>
 0.00 :        95b2:       test   %rax,%rax
 0.00 :        95b5:       mov    0x18(%rsp),%rsi
 0.00 :        95ba:       mov    (%rsp),%r8
 0.00 :        95be:       mov    0x10(%rsp),%r9
 0.00 :        95c3:       movzbl 0x8(%rsp),%r11d
 0.00 :        95c9:       je     9568 <do_lookup_x+0x808>
 0.00 :        95cb:       jmpq   8f46 <do_lookup_x+0x1e6>
 0.00 :        95d0:       mov    0x78(%rsp),%rcx
 0.00 :        95d5:       movzbl (%rcx),%edx
```

```
    0.00 :          95d8:       test   %rdx,%rdx
    0.00 :          95db:       je     96c6 <do_lookup_x+0x966>
    0.00 :          95e1:       movzbl 0x1(%rcx),%r10d
    0.00 :          95e6:       test   %r10b,%r10b
    0.00 :          95e9:       je     96c6 <do_lookup_x+0x966>
    0.00 :          95ef:       shl    $0x4,%rdx
    0.00 :          95f3:       add    %r10,%rdx
    0.00 :          95f6:       movzbl 0x2(%rcx),%r10d
    0.00 :          95fb:       test   %r10b,%r10b
    0.00 :          95fe:       je     96c6 <do_lookup_x+0x966>
    0.00 :          9604:       shl    $0x4,%rdx
    0.00 :          9608:       add    %r10,%rdx
    0.00 :          960b:       movzbl 0x3(%rcx),%r10d
    0.00 :          9610:       test   %r10b,%r10b
    0.00 :          9613:       je     96c6 <do_lookup_x+0x966>
    0.00 :          9619:       shl    $0x4,%rdx
    0.00 :          961d:       add    %r10,%rdx
    0.00 :          9620:       movzbl 0x4(%rcx),%r10d
    0.00 :          9625:       test   %r10b,%r10b
    0.00 :          9628:       je     96c6 <do_lookup_x+0x966>
    0.00 :          962e:       mov    0x38(%rsp),%rsi
    0.00 :          9633:       shl    $0x4,%rdx
    0.00 :          9637:       add    $0x5,%rcx
    0.00 :          963b:       add    %r10,%rdx
    0.00 :          963e:       movzbl (%rcx),%r10d
    0.00 :          9642:       test   %r10b,%r10b
    0.00 :          9645:       je     96bb <do_lookup_x+0x95b>
    0.00 :          9647:       shl    $0x4,%rdx
    0.00 :          964b:       add    $0x1,%rcx
    0.00 :          964f:       add    %rdx,%r10
    0.00 :          9652:       mov    %r10,%rdx
    0.00 :          9655:       and    $0xf0000000,%edx
    0.00 :          965b:       shr    $0x18,%rdx
    0.00 :          965f:       xor    %r10,%rdx
    0.00 :          9662:       jmp    963e <do_lookup_x+0x8de>
    0.00 :          9664:       testb  $0x8,0x218536(%rip)        # 221ba1 <_rtld_global_ro+0x1>
    0.00 :          966b:       je     969c <do_lookup_x+0x93c>
    0.00 :          966d:       mov    0x38(%rsp),%rdi
    0.00 :          9672:       callq  *0x219890(%rip)        # 222f08 <_rtld_global+0xf08>
    0.00 :          9678:       jmpq   8f65 <do_lookup_x+0x205>
    0.00 :          967d:       lea    0x11d3d(%rip),%rcx        # 1b3c1 <__PRETTY_FUNCTION__.9401>
    0.00 :          9684:       lea    0x11c52(%rip),%rsi        # 1b2dd <__PRETTY_FUNCTION__.9569+0xf>
    0.00 :          968b:       lea    0x13836(%rip),%rdi        # 1cec8 <__PRETTY_FUNCTION__.4816+0x135f>
    0.00 :          9692:       mov    $0x197,%edx
    0.00 :          9697:       callq  15980 <__GI___assert_fail>
    0.00 :          969c:       lea    0x11d1e(%rip),%rcx        # 1b3c1 <__PRETTY_FUNCTION__.9401>
    0.00 :          96a3:       lea    0x11c33(%rip),%rsi        # 1b2dd <__PRETTY_FUNCTION__.9569+0xf>
    0.00 :          96aa:       lea    0x13acf(%rip),%rdi        # 1d180 <__PRETTY_FUNCTION__.4816+0x1617>
    0.00 :          96b1:       mov    $0x1a2,%edx
    0.00 :          96b6:       callq  15980 <__GI___assert_fail>
    0.00 :          96bb:       mov    %rsi,0x38(%rsp)
    0.00 :          96c0:       and    $0xffffffff,%edx
    0.00 :          96c6:       mov    %edx,%edx
    0.00 :          96c8:       mov    %rdx,(%r15)
    0.00 :          96cb:       jmpq   9537 <do_lookup_x+0x7d7>
 Percent |       Source code & Disassembly of [jbd2]
-----------------------------------------------
 Percent |       Source code & Disassembly of [ext4]
-----------------------------------------------
 Percent |       Source code & Disassembly of [ext4]
-----------------------------------------------
 Percent |       Source code & Disassembly of [ext4]
-----------------------------------------------
 Percent |       Source code & Disassembly of [ext4]
```

```
-----------------------------------------------
 Percent |        Source code & Disassembly of libz.so.1.2.7
-----------------------------------------------
         :
         :
         :
         :        Disassembly of section .text:
         :
         :        0000000000004640 <deflate>:
    0.00 :          4640:       push   %r14
    0.00 :          4642:       test   %rdi,%rdi
    0.00 :          4645:       push   %r13
    0.00 :          4647:       push   %r12
    0.00 :          4649:       push   %rbp
    0.00 :          464a:       mov    %rdi,%rbp
    0.00 :          464d:       push   %rbx
    0.00 :          464e:       je     5224 <deflate+0xbe4>
    0.00 :          4654:       mov    0x38(%rdi),%rbx
    0.00 :          4658:       test   %rbx,%rbx
    0.00 :          465b:       je     5224 <deflate+0xbe4>
    0.00 :          4661:       cmp    $0x5,%esi
    0.00 :          4664:       mov    %esi,%r12d
    0.00 :          4667:       ja     5224 <deflate+0xbe4>
    0.00 :          466d:       cmpq   $0x0,0x18(%rdi)
    0.00 :          4672:       je     47ea <deflate+0x1aa>
    0.00 :          4678:       cmpq   $0x0,(%rdi)
    0.00 :          467c:       je     5070 <deflate+0xa30>
    0.00 :          4682:       mov    0x8(%rbx),%eax
    0.00 :          4685:       cmp    $0x29a,%eax
    0.00 :          468a:       je     47e0 <deflate+0x1a0>
    0.00 :          4690:       mov    0x20(%rbp),%r9d
    0.00 :          4694:       test   %r9d,%r9d
    0.00 :          4697:       je     4c77 <deflate+0x637>
    0.00 :          469d:       cmp    $0x2a,%eax
    0.00 :          46a0:       mov    0x40(%rbx),%r13d
    0.00 :          46a4:       mov    %rbp,(%rbx)
    0.00 :          46a7:       mov    %r12d,0x40(%rbx)
    0.00 :          46ab:       je     4940 <deflate+0x300>
    0.00 :          46b1:       cmp    $0x45,%eax
    0.00 :          46b4:       je     4a26 <deflate+0x3e6>
    0.00 :          46ba:       cmp    $0x49,%eax
    0.00 :          46bd:       mov    0x28(%rbx),%edx
    0.00 :          46c0:       je     4cc8 <deflate+0x688>
    0.00 :          46c6:       cmp    $0x5b,%eax
    0.00 :          46c9:       je     5060 <deflate+0xa20>
    0.00 :          46cf:       cmp    $0x67,%eax
    0.00 :          46d2:       je     5160 <deflate+0xb20>
    0.00 :          46d8:       test   %edx,%edx
    0.00 :          46da:       jne    4808 <deflate+0x1c8>
    0.00 :          46e0:       mov    0x8(%rbp),%eax
    0.00 :          46e3:       test   %eax,%eax
    0.00 :          46e5:       jne    4c60 <deflate+0x620>
    0.00 :          46eb:       xor    %eax,%eax
    0.00 :          46ed:       cmp    $0x5,%r12d
    0.00 :          46f1:       lea    (%r12,%r12,1),%ecx
    0.00 :          46f5:       sete   %al
    0.00 :          46f8:       lea    0x0(%r13,%r13,1),%edx
    0.00 :          46fd:       lea    (%rax,%rax,8),%eax
    0.00 :          4700:       sub    %eax,%ecx
    0.00 :          4702:       xor    %eax,%eax
    0.00 :          4704:       cmp    $0x5,%r13d
    0.00 :          4708:       setge  %al
    0.00 :          470b:       lea    (%rax,%rax,8),%eax
    0.00 :          470e:       sub    %eax,%edx
```

```
 0.00 :        4710:       cmp    %edx,%ecx
 0.00 :        4712:       jg     471e <deflate+0xde>
 0.00 :        4714:       cmp    $0x4,%r12d
 0.00 :        4718:       jne    4c77 <deflate+0x637>
 0.00 :        471e:       mov    0x8(%rbx),%edx
 0.00 :        4721:       nopl   0x0(%rax)
 0.00 :        4728:       mov    0xa4(%rbx),%eax
 0.00 :        472e:       test   %eax,%eax
 0.00 :        4730:       jne    4835 <deflate+0x1f5>
 0.00 :        4736:       test   %r12d,%r12d
 0.00 :        4739:       je     488b <deflate+0x24b>
 0.00 :        473f:       cmp    $0x29a,%edx
 0.00 :        4745:       jne    4835 <deflate+0x1f5>
 0.00 :        474b:       nopl   0x0(%rax,%rax,1)
 0.00 :        4750:       cmp    $0x4,%r12d
 0.00 :        4754:       jne    488b <deflate+0x24b>
 0.00 :        475a:       mov    0x2c(%rbx),%eax
 0.00 :        475d:       test   %eax,%eax
 0.00 :        475f:       jle    550d <deflate+0xecd>
 0.00 :        4765:       cmp    $0x2,%eax
 0.00 :        4768:       je     5290 <deflate+0xc50>
 0.00 :        476e:       mov    0x60(%rbp),%rdx
 0.00 :        4772:       mov    0x28(%rbx),%eax
 0.00 :        4775:       mov    0x10(%rbx),%rsi
 0.00 :        4779:       shr    $0x10,%rdx
 0.00 :        477d:       mov    %eax,%ecx
 0.00 :        477f:       mov    %edx,%edi
 0.00 :        4781:       shr    $0x8,%edi
 0.00 :        4784:       mov    %dil,(%rsi,%rcx,1)
 0.00 :        4788:       mov    0x10(%rbx),%rsi
 0.00 :        478c:       lea    0x1(%rax),%ecx
 0.00 :        478f:       mov    %dl,(%rsi,%rcx,1)
 0.00 :        4792:       movzwl 0x60(%rbp),%edx
 0.00 :        4796:       lea    0x2(%rax),%ecx
 0.00 :        4799:       mov    0x10(%rbx),%rsi
 0.00 :        479d:       mov    %edx,%edi
 0.00 :        479f:       shr    $0x8,%edi
 0.00 :        47a2:       mov    %dil,(%rsi,%rcx,1)
 0.00 :        47a6:       mov    0x10(%rbx),%rsi
 0.00 :        47aa:       lea    0x3(%rax),%ecx
 0.00 :        47ad:       add    $0x4,%eax
 0.00 :        47b0:       mov    %dl,(%rsi,%rcx,1)
 0.00 :        47b3:       mov    %eax,0x28(%rbx)
 0.00 :        47b6:       mov    %rbp,%rdi
 0.00 :        47b9:       callq  3380 <crc32_combine64+0x5a0>
 0.00 :        47be:       mov    0x2c(%rbx),%eax
 0.00 :        47c1:       test   %eax,%eax
 0.00 :        47c3:       jle    47ca <deflate+0x18a>
 0.00 :        47c5:       neg    %eax
 0.00 :        47c7:       mov    %eax,0x2c(%rbx)
 0.00 :        47ca:       mov    0x28(%rbx),%r8d
 0.00 :        47ce:       xor    %eax,%eax
 0.00 :        47d0:       pop    %rbx
 0.00 :        47d1:       pop    %rbp
 0.00 :        47d2:       pop    %r12
 0.00 :        47d4:       test   %r8d,%r8d
 0.00 :        47d7:       pop    %r13
 0.00 :        47d9:       sete   %al
 0.00 :        47dc:       pop    %r14
 0.00 :        47de:       retq
 0.00 :        47df:       nop
 0.00 :        47e0:       cmp    $0x4,%r12d
 0.00 :        47e4:       je     4690 <deflate+0x50>
 0.00 :        47ea:       mov    0x2107c7(%rip),%rax        # 214fb8 <gzclose_w+0x206938>
```

```
  0.00 :        47f1:       mov    0x20(%rax),%rax
  0.00 :        47f5:       mov    %rax,0x30(%rbp)
  0.00 :        47f9:       mov    $0xfffffffe,%eax
  0.00 :        47fe:       jmpq   488d <deflate+0x24d>
  0.00 :        4803:       nopl   0x0(%rax,%rax,1)
  0.00 :        4808:       mov    %rbp,%rdi
  0.00 :        480b:       callq  3380 <crc32_combine64+0x5a0>
  0.00 :        4810:       mov    0x20(%rbp),%eax
  0.00 :        4813:       test   %eax,%eax
  0.00 :        4815:       je     4930 <deflate+0x2f0>
100.00 :        481b:       mov    0x8(%rbx),%edx
  0.00 :        481e:       mov    0x8(%rbp),%eax
  0.00 :        4821:       cmp    $0x29a,%edx
  0.00 :        4827:       je     4c6f <deflate+0x62f>
  0.00 :        482d:       test   %eax,%eax
  0.00 :        482f:       je     4728 <deflate+0xe8>
  0.00 :        4835:       mov    0xb8(%rbx),%eax
  0.00 :        483b:       cmp    $0x2,%eax
  0.00 :        483e:       je     4b0e <deflate+0x4ce>
  0.00 :        4844:       cmp    $0x3,%eax
  0.00 :        4847:       je     4db0 <deflate+0x770>
  0.00 :        484d:       movslq 0xb4(%rbx),%rax
  0.00 :        4854:       lea    0x210465(%rip),%rdx        # 214cc0 <gzclose_w+0x206640>
  0.00 :        485b:       mov    %r12d,%esi
  0.00 :        485e:       mov    %rbx,%rdi
  0.00 :        4861:       shl    $0x4,%rax
  0.00 :        4865:       callq  *0x8(%rdx,%rax,1)
  0.00 :        4869:       mov    %eax,%edx
  0.00 :        486b:       lea    -0x2(%rax),%ecx
  0.00 :        486e:       and    $0xfffffffd,%edx
  0.00 :        4871:       cmp    $0x1,%ecx
  0.00 :        4874:       jbe    4da0 <deflate+0x760>
  0.00 :        487a:       test   %edx,%edx
  0.00 :        487c:       jne    48a0 <deflate+0x260>
  0.00 :        487e:       mov    0x20(%rbp),%r11d
  0.00 :        4882:       test   %r11d,%r11d
  0.00 :        4885:       je     4930 <deflate+0x2f0>
  0.00 :        488b:       xor    %eax,%eax
  0.00 :        488d:       pop    %rbx
  0.00 :        488e:       pop    %rbp
  0.00 :        488f:       pop    %r12
  0.00 :        4891:       pop    %r13
  0.00 :        4893:       pop    %r14
  0.00 :        4895:       retq
  0.00 :        4896:       nopw   %cs:0x0(%rax,%rax,1)
  0.00 :        48a0:       cmp    $0x1,%eax
  0.00 :        48a3:       jne    4750 <deflate+0x110>
  0.00 :        48a9:       cmp    $0x1,%r12d
  0.00 :        48ad:       je     552d <deflate+0xeed>
  0.00 :        48b3:       cmp    $0x5,%r12d
  0.00 :        48b7:       je     4913 <deflate+0x2d3>
  0.00 :        48b9:       xor    %ecx,%ecx
  0.00 :        48bb:       xor    %edx,%edx
  0.00 :        48bd:       xor    %esi,%esi
  0.00 :        48bf:       mov    %rbx,%rdi
  0.00 :        48c2:       callq  bc30 <inflateMark+0x19d0>
  0.00 :        48c7:       cmp    $0x3,%r12d
  0.00 :        48cb:       jne    4913 <deflate+0x2d3>
  0.00 :        48cd:       mov    0x74(%rbx),%eax
  0.00 :        48d0:       mov    0x68(%rbx),%rdi
  0.00 :        48d4:       xor    %esi,%esi
  0.00 :        48d6:       lea    -0x1(%rax),%eax
  0.00 :        48d9:       lea    (%rax,%rax,1),%rdx
  0.00 :        48dd:       movw   $0x0,(%rdi,%rax,2)
```

```
 0.00 :        48e3:       callq  20f0 <memset@plt>
 0.00 :        48e8:       mov    0xa4(%rbx),%r10d
 0.00 :        48ef:       test   %r10d,%r10d
 0.00 :        48f2:       jne    4913 <deflate+0x2d3>
 0.00 :        48f4:       movl   $0x0,0x9c(%rbx)
 0.00 :        48fe:       movq   $0x0,0x88(%rbx)
 0.00 :        4909:       movl   $0x0,0x171c(%rbx)
 0.00 :        4913:       mov    %rbp,%rdi
 0.00 :        4916:       callq  3380 <crc32_combine64+0x5a0>
 0.00 :        491b:       mov    0x20(%rbp),%r9d
 0.00 :        491f:       test   %r9d,%r9d
 0.00 :        4922:       jne    4750 <deflate+0x110>
 0.00 :        4928:       nopl   0x0(%rax,%rax,1)
 0.00 :        4930:       movl   $0xffffffff,0x40(%rbx)
 0.00 :        4937:       jmpq   488b <deflate+0x24b>
 0.00 :        493c:       nopl   0x0(%rax)
 0.00 :        4940:       cmpl   $0x2,0x2c(%rbx)
 0.00 :        4944:       je     5327 <deflate+0xce7>
 0.00 :        494a:       mov    0x48(%rbx),%eax
 0.00 :        494d:       shl    $0xc,%eax
 0.00 :        4950:       lea    -0x7800(%rax),%ecx
 0.00 :        4956:       xor    %eax,%eax
 0.00 :        4958:       cmpl   $0x1,0xb8(%rbx)
 0.00 :        495f:       jle    4c34 <deflate+0x5f4>
 0.00 :        4965:       mov    0x9c(%rbx),%r14d
 0.00 :        496c:       or     %eax,%ecx
 0.00 :        496e:       mov    $0x8421085,%edx
 0.00 :        4973:       mov    %ecx,%eax
 0.00 :        4975:       mov    0x10(%rbx),%rsi
 0.00 :        4979:       movl   $0x71,0x8(%rbx)
 0.00 :        4980:       or     $0x20,%eax
 0.00 :        4983:       test   %r14d,%r14d
 0.00 :        4986:       cmovne %eax,%ecx
 0.00 :        4989:       mov    %ecx,%eax
 0.00 :        498b:       mul    %edx
 0.00 :        498d:       sub    %edx,%ecx
 0.00 :        498f:       shr    %ecx
 0.00 :        4991:       add    %edx,%ecx
 0.00 :        4993:       shr    $0x4,%ecx
 0.00 :        4996:       mov    %ecx,%eax
 0.00 :        4998:       shl    $0x5,%eax
 0.00 :        499b:       sub    %ecx,%eax
 0.00 :        499d:       lea    0x1f(%rax),%edx
 0.00 :        49a0:       mov    0x28(%rbx),%eax
 0.00 :        49a3:       mov    %edx,%edi
 0.00 :        49a5:       mov    %eax,%ecx
 0.00 :        49a7:       shr    $0x8,%edi
 0.00 :        49aa:       mov    %dil,(%rsi,%rcx,1)
 0.00 :        49ae:       mov    0x10(%rbx),%rsi
 0.00 :        49b2:       lea    0x1(%rax),%ecx
 0.00 :        49b5:       mov    %dl,(%rsi,%rcx,1)
 0.00 :        49b8:       mov    0x9c(%rbx),%ecx
 0.00 :        49be:       lea    0x2(%rax),%edx
 0.00 :        49c1:       mov    %edx,0x28(%rbx)
 0.00 :        49c4:       test   %ecx,%ecx
 0.00 :        49c6:       je     4a0b <deflate+0x3cb>
 0.00 :        49c8:       mov    0x60(%rbp),%rcx
 0.00 :        49cc:       mov    0x10(%rbx),%rsi
 0.00 :        49d0:       shr    $0x10,%rcx
 0.00 :        49d4:       mov    %ecx,%edi
 0.00 :        49d6:       shr    $0x8,%edi
 0.00 :        49d9:       mov    %dil,(%rsi,%rdx,1)
 0.00 :        49dd:       mov    0x10(%rbx),%rsi
 0.00 :        49e1:       lea    0x3(%rax),%edx
```

```
 0.00 :        49e4:       mov    %cl,(%rsi,%rdx,1)
 0.00 :        49e7:       movzwl 0x60(%rbp),%edx
 0.00 :        49eb:       lea    0x4(%rax),%ecx
 0.00 :        49ee:       mov    0x10(%rbx),%rsi
 0.00 :        49f2:       mov    %edx,%edi
 0.00 :        49f4:       shr    $0x8,%edi
 0.00 :        49f7:       mov    %dil,(%rsi,%rcx,1)
 0.00 :        49fb:       mov    0x10(%rbx),%rsi
 0.00 :        49ff:       lea    0x5(%rax),%ecx
 0.00 :        4a02:       add    $0x6,%eax
 0.00 :        4a05:       mov    %dl,(%rsi,%rcx,1)
 0.00 :        4a08:       mov    %eax,0x28(%rbx)
 0.00 :        4a0b:       xor    %edx,%edx
 0.00 :        4a0d:       xor    %esi,%esi
 0.00 :        4a0f:       xor    %edi,%edi
 0.00 :        4a11:       callq  21a0 <adler32@plt>
 0.00 :        4a16:       mov    %rax,0x60(%rbp)
 0.00 :        4a1a:       mov    0x8(%rbx),%eax
 0.00 :        4a1d:       cmp    $0x45,%eax
 0.00 :        4a20:       jne    46ba <deflate+0x7a>
 0.00 :        4a26:       mov    0x30(%rbx),%rsi
 0.00 :        4a2a:       mov    0x28(%rbx),%r8d
 0.00 :        4a2e:       mov    %rsi,%rax
 0.00 :        4a31:       cmpq   $0x0,0x18(%rsi)
 0.00 :        4a36:       je     5249 <deflate+0xc09>
 0.00 :        4a3c:       movzwl 0x20(%rsi),%edx
 0.00 :        4a40:       cmp    %edx,0x38(%rbx)
 0.00 :        4a43:       jae    4c90 <deflate+0x650>
 0.00 :        4a49:       mov    %r8d,%edx
 0.00 :        4a4c:       jmp    4a85 <deflate+0x445>
 0.00 :        4a4e:       xchg   %ax,%ax
 0.00 :        4a50:       mov    0x38(%rbx),%ecx
 0.00 :        4a53:       mov    0x18(%rsi),%rsi
 0.00 :        4a57:       add    $0x1,%edx
 0.00 :        4a5a:       movzbl (%rsi,%rcx,1),%esi
 0.00 :        4a5e:       mov    0x10(%rbx),%rcx
 0.00 :        4a62:       mov    %sil,(%rcx,%rax,1)
 0.00 :        4a66:       mov    0x30(%rbx),%rax
 0.00 :        4a6a:       mov    0x38(%rbx),%ecx
 0.00 :        4a6d:       mov    %edx,0x28(%rbx)
 0.00 :        4a70:       movzwl 0x20(%rax),%edi
 0.00 :        4a74:       mov    %rax,%rsi
 0.00 :        4a77:       add    $0x1,%ecx
 0.00 :        4a7a:       mov    %ecx,0x38(%rbx)
 0.00 :        4a7d:       cmp    %ecx,%edi
 0.00 :        4a7f:       jbe    4c98 <deflate+0x658>
 0.00 :        4a85:       mov    %edx,%eax
 0.00 :        4a87:       cmp    0x18(%rbx),%rax
 0.00 :        4a8b:       jne    4a50 <deflate+0x410>
 0.00 :        4a8d:       mov    0x44(%rsi),%r11d
 0.00 :        4a91:       test   %r11d,%r11d
 0.00 :        4a94:       je     4a9f <deflate+0x45f>
 0.00 :        4a96:       cmp    %edx,%r8d
 0.00 :        4a99:       jb     5088 <deflate+0xa48>
 0.00 :        4a9f:       mov    %rbp,%rdi
 0.00 :        4aa2:       callq  3380 <crc32_combine64+0x5a0>
 0.00 :        4aa7:       mov    0x28(%rbx),%r8d
 0.00 :        4aab:       mov    %r8d,%eax
 0.00 :        4aae:       cmp    0x18(%rbx),%rax
 0.00 :        4ab2:       mov    %r8d,%edx
 0.00 :        4ab5:       je     5240 <deflate+0xc00>
 0.00 :        4abb:       mov    %r8d,%edx
 0.00 :        4abe:       mov    0x30(%rbx),%rsi
 0.00 :        4ac2:       jmp    4a50 <deflate+0x410>
```

```
 0.00 :        4ac4:       nopl   0x0(%rax)
 0.00 :        4ac8:       mov    0x88(%rbx),%rax
 0.00 :        4acf:       mov    %ecx,%edx
 0.00 :        4ad1:       xor    %esi,%esi
 0.00 :        4ad3:       sub    %rax,%rdx
 0.00 :        4ad6:       test   %rax,%rax
 0.00 :        4ad9:       js     4ae1 <deflate+0x4a1>
 0.00 :        4adb:       mov    %eax,%esi
 0.00 :        4add:       add    0x50(%rbx),%rsi
 0.00 :        4ae1:       xor    %ecx,%ecx
 0.00 :        4ae3:       mov    %rbx,%rdi
 0.00 :        4ae6:       callq  be40 <inflateMark+0x1be0>
 0.00 :        4aeb:       mov    0x9c(%rbx),%eax
 0.00 :        4af1:       mov    (%rbx),%rdi
 0.00 :        4af4:       mov    %rax,0x88(%rbx)
 0.00 :        4afb:       callq  3380 <crc32_combine64+0x5a0>
 0.00 :        4b00:       mov    (%rbx),%rax
 0.00 :        4b03:       mov    0x20(%rax),%eax
 0.00 :        4b06:       test   %eax,%eax
 0.00 :        4b08:       je     487e <deflate+0x23e>
 0.00 :        4b0e:       mov    0xa4(%rbx),%eax
 0.00 :        4b14:       jmpq   4b98 <deflate+0x558>
 0.00 :        4b19:       nopl   0x0(%rax)
 0.00 :        4b20:       mov    0x9c(%rbx),%eax
 0.00 :        4b26:       mov    0x50(%rbx),%rdx
 0.00 :        4b2a:       movl   $0x0,0x90(%rbx)
 0.00 :        4b34:       mov    0x1700(%rbx),%rsi
 0.00 :        4b3b:       movzbl (%rdx,%rax,1),%eax
 0.00 :        4b3f:       mov    0x16fc(%rbx),%edx
 0.00 :        4b45:       mov    %edx,%ecx
 0.00 :        4b47:       add    $0x1,%edx
 0.00 :        4b4a:       movw   $0x0,(%rsi,%rcx,2)
 0.00 :        4b50:       mov    0x16f0(%rbx),%rsi
 0.00 :        4b57:       mov    %al,(%rsi,%rcx,1)
 0.00 :        4b5a:       mov    %edx,0x16fc(%rbx)
 0.00 :        4b60:       addw   $0x1,0xc4(%rbx,%rax,4)
 0.00 :        4b69:       mov    0x16f8(%rbx),%esi
 0.00 :        4b6f:       mov    0xa4(%rbx),%eax
 0.00 :        4b75:       mov    0x9c(%rbx),%ecx
 0.00 :        4b7b:       sub    $0x1,%esi
 0.00 :        4b7e:       sub    $0x1,%eax
 0.00 :        4b81:       add    $0x1,%ecx
 0.00 :        4b84:       cmp    %esi,%edx
 0.00 :        4b86:       mov    %eax,0xa4(%rbx)
 0.00 :        4b8c:       mov    %ecx,0x9c(%rbx)
 0.00 :        4b92:       je     4ac8 <deflate+0x488>
 0.00 :        4b98:       test   %eax,%eax
 0.00 :        4b9a:       jne    4b20 <deflate+0x4e0>
 0.00 :        4b9c:       mov    %rbx,%rdi
 0.00 :        4b9f:       callq  3040 <crc32_combine64+0x260>
 0.00 :        4ba4:       mov    0xa4(%rbx),%eax
 0.00 :        4baa:       test   %eax,%eax
 0.00 :        4bac:       jne    4b20 <deflate+0x4e0>
 0.00 :        4bb2:       test   %r12d,%r12d
 0.00 :        4bb5:       je     487e <deflate+0x23e>
 0.00 :        4bbb:       cmp    $0x4,%r12d
 0.00 :        4bbf:       movl   $0x0,0x171c(%rbx)
 0.00 :        4bc9:       je     4d48 <deflate+0x708>
 0.00 :        4bcf:       mov    0x16fc(%rbx),%r14d
 0.00 :        4bd6:       test   %r14d,%r14d
 0.00 :        4bd9:       je     48a9 <deflate+0x269>
 0.00 :        4bdf:       mov    0x9c(%rbx),%edx
 0.00 :        4be5:       mov    0x88(%rbx),%rax
 0.00 :        4bec:       xor    %esi,%esi
```

```
 0.00 :         4bee:       sub    %rax,%rdx
 0.00 :         4bf1:       test   %rax,%rax
 0.00 :         4bf4:       js     4bfc <deflate+0x5bc>
 0.00 :         4bf6:       mov    %eax,%esi
 0.00 :         4bf8:       add    0x50(%rbx),%rsi
 0.00 :         4bfc:       xor    %ecx,%ecx
 0.00 :         4bfe:       mov    %rbx,%rdi
 0.00 :         4c01:       callq  be40 <inflateMark+0x1be0>
 0.00 :         4c06:       mov    0x9c(%rbx),%ecx
 0.00 :         4c0c:       mov    (%rbx),%rdi
 0.00 :         4c0f:       mov    %rcx,0x88(%rbx)
 0.00 :         4c16:       callq  3380 <crc32_combine64+0x5a0>
 0.00 :         4c1b:       mov    (%rbx),%rax
 0.00 :         4c1e:       mov    0x20(%rax),%r13d
 0.00 :         4c22:       xor    %eax,%eax
 0.00 :         4c24:       test   %r13d,%r13d
 0.00 :         4c27:       setne  %al
 0.00 :         4c2a:       lea    -0x2(%rax),%ecx
 0.00 :         4c2d:       mov    %eax,%edx
 0.00 :         4c2f:       jmpq   4871 <deflate+0x231>
 0.00 :         4c34:       mov    0xb4(%rbx),%edx
 0.00 :         4c3a:       cmp    $0x1,%edx
 0.00 :         4c3d:       jle    4965 <deflate+0x325>
 0.00 :         4c43:       cmp    $0x5,%edx
 0.00 :         4c46:       mov    $0x40,%al
 0.00 :         4c48:       jle    4965 <deflate+0x325>
 0.00 :         4c4e:       cmp    $0x6,%edx
 0.00 :         4c51:       mov    $0x80,%al
 0.00 :         4c53:       mov    $0xc0,%edx
 0.00 :         4c58:       cmovne %edx,%eax
 0.00 :         4c5b:       jmpq   4965 <deflate+0x325>
 0.00 :         4c60:       mov    0x8(%rbx),%edx
 0.00 :         4c63:       cmp    $0x29a,%edx
 0.00 :         4c69:       jne    4835 <deflate+0x1f5>
 0.00 :         4c6f:       test   %eax,%eax
 0.00 :         4c71:       je     4728 <deflate+0xe8>
 0.00 :         4c77:       mov    0x21033a(%rip),%rax        # 214fb8 <gzclose_w+0x206938>
 0.00 :         4c7e:       mov    0x38(%rax),%rax
 0.00 :         4c82:       mov    %rax,0x30(%rbp)
 0.00 :         4c86:       mov    $0xfffffffb,%eax
 0.00 :         4c8b:       jmpq   488d <deflate+0x24d>
 0.00 :         4c90:       mov    0x28(%rbx),%edx
 0.00 :         4c93:       nopl   0x0(%rax,%rax,1)
 0.00 :         4c98:       mov    0x44(%rax),%r10d
 0.00 :         4c9c:       test   %r10d,%r10d
 0.00 :         4c9f:       je     4caa <deflate+0x66a>
 0.00 :         4ca1:       cmp    %edx,%r8d
 0.00 :         4ca4:       jb     54ea <deflate+0xeaa>
 0.00 :         4caa:       mov    0x20(%rax),%ecx
 0.00 :         4cad:       cmp    %ecx,0x38(%rbx)
 0.00 :         4cb0:       je     51d9 <deflate+0xb99>
 0.00 :         4cb6:       mov    0x8(%rbx),%eax
 0.00 :         4cb9:       cmp    $0x49,%eax
 0.00 :         4cbc:       jne    46c6 <deflate+0x86>
 0.00 :         4cc2:       nopw   0x0(%rax,%rax,1)
 0.00 :         4cc8:       mov    0x30(%rbx),%rax
 0.00 :         4ccc:       cmpq   $0x0,0x28(%rax)
 0.00 :         4cd1:       je     5264 <deflate+0xc24>
 0.00 :         4cd7:       mov    %edx,%r8d
 0.00 :         4cda:       jmp    4d0c <deflate+0x6cc>
 0.00 :         4cdc:       nopl   0x0(%rax)
 0.00 :         4ce0:       mov    0x38(%rbx),%esi
 0.00 :         4ce3:       mov    0x28(%rax),%rax
 0.00 :         4ce7:       add    $0x1,%edx
```

```
 0.00 :         4cea:       mov    %esi,%edi
 0.00 :         4cec:       add    $0x1,%esi
 0.00 :         4cef:       movzbl (%rax,%rdi,1),%eax
 0.00 :         4cf3:       mov    %esi,0x38(%rbx)
 0.00 :         4cf6:       mov    0x10(%rbx),%rsi
 0.00 :         4cfa:       test   %al,%al
 0.00 :         4cfc:       mov    %al,(%rsi,%rcx,1)
 0.00 :         4cff:       mov    %edx,0x28(%rbx)
 0.00 :         4d02:       je     5038 <deflate+0x9f8>
 0.00 :         4d08:       mov    0x30(%rbx),%rax
 0.00 :         4d0c:       mov    %edx,%ecx
 0.00 :         4d0e:       cmp    0x18(%rbx),%rcx
 0.00 :         4d12:       jne    4ce0 <deflate+0x6a0>
 0.00 :         4d14:       mov    0x44(%rax),%r9d
 0.00 :         4d18:       test   %r9d,%r9d
 0.00 :         4d1b:       je     4d26 <deflate+0x6e6>
 0.00 :         4d1d:       cmp    %edx,%r8d
 0.00 :         4d20:       jb     5208 <deflate+0xbc8>
 0.00 :         4d26:       mov    %rbp,%rdi
 0.00 :         4d29:       callq  3380 <crc32_combine64+0x5a0>
 0.00 :         4d2e:       mov    0x28(%rbx),%r8d
 0.00 :         4d32:       mov    %r8d,%ecx
 0.00 :         4d35:       cmp    0x18(%rbx),%rcx
 0.00 :         4d39:       mov    %r8d,%edx
 0.00 :         4d3c:       je     5517 <deflate+0xed7>
 0.00 :         4d42:       mov    0x30(%rbx),%rax
 0.00 :         4d46:       jmp    4ce0 <deflate+0x6a0>
 0.00 :         4d48:       mov    0x9c(%rbx),%edx
 0.00 :         4d4e:       mov    0x88(%rbx),%rax
 0.00 :         4d55:       xor    %esi,%esi
 0.00 :         4d57:       sub    %rax,%rdx
 0.00 :         4d5a:       test   %rax,%rax
 0.00 :         4d5d:       js     4d65 <deflate+0x725>
 0.00 :         4d5f:       mov    %eax,%esi
 0.00 :         4d61:       add    0x50(%rbx),%rsi
 0.00 :         4d65:       mov    $0x1,%ecx
 0.00 :         4d6a:       mov    %rbx,%rdi
 0.00 :         4d6d:       callq  be40 <inflateMark+0x1be0>
 0.00 :         4d72:       mov    0x9c(%rbx),%edx
 0.00 :         4d78:       mov    (%rbx),%rdi
 0.00 :         4d7b:       mov    %rdx,0x88(%rbx)
 0.00 :         4d82:       callq  3380 <crc32_combine64+0x5a0>
 0.00 :         4d87:       mov    (%rbx),%rax
 0.00 :         4d8a:       cmpl   $0x1,0x20(%rax)
 0.00 :         4d8e:       sbb    %edx,%edx
 0.00 :         4d90:       add    $0x1,%edx
 0.00 :         4d93:       cmpl   $0x1,0x20(%rax)
 0.00 :         4d97:       sbb    %eax,%eax
 0.00 :         4d99:       add    $0x3,%eax
 0.00 :         4d9c:       nopl   0x0(%rax)
 0.00 :         4da0:       movl   $0x29a,0x8(%rbx)
 0.00 :         4da7:       jmpq   487a <deflate+0x23a>
 0.00 :         4dac:       nopl   0x0(%rax)
 0.00 :         4db0:       mov    $0x102,%r13d
 0.00 :         4db6:       movzbl 0xd082(%rip),%r14d        # 11e40 <gzclose_w+0x37c0>
 0.00 :         4dbe:       jmpq   4e59 <deflate+0x819>
 0.00 :         4dc3:       nopl   0x0(%rax,%rax,1)
 0.00 :         4dc8:       movl   $0x0,0x90(%rbx)
 0.00 :         4dd2:       mov    0x9c(%rbx),%ecx
 0.00 :         4dd8:       mov    0x50(%rbx),%rdx
 0.00 :         4ddc:       test   %ecx,%ecx
 0.00 :         4dde:       je     4df7 <deflate+0x7b7>
 0.00 :         4de0:       mov    %ecx,%r9d
 0.00 :         4de3:       lea    -0x1(%rdx,%r9,1),%rsi
```

```
 0.00 :        4de8:       movzbl (%rsi),%ecx
 0.00 :        4deb:       movzbl (%rsi),%edi
 0.00 :        4dee:       cmp    %cl,0x1(%rsi)
 0.00 :        4df1:       je     4eb0 <deflate+0x870>
 0.00 :        4df7:       mov    0x9c(%rbx),%eax
 0.00 :        4dfd:       mov    0x1700(%rbx),%rsi
 0.00 :        4e04:       movzbl (%rdx,%rax,1),%edx
 0.00 :        4e08:       mov    0x16fc(%rbx),%eax
 0.00 :        4e0e:       mov    %eax,%ecx
 0.00 :        4e10:       add    $0x1,%eax
 0.00 :        4e13:       movw   $0x0,(%rsi,%rcx,2)
 0.00 :        4e19:       mov    0x16f0(%rbx),%rsi
 0.00 :        4e20:       mov    %dl,(%rsi,%rcx,1)
 0.00 :        4e23:       mov    %eax,0x16fc(%rbx)
 0.00 :        4e29:       addw   $0x1,0xc4(%rbx,%rdx,4)
 0.00 :        4e32:       mov    0x16f8(%rbx),%edx
 0.00 :        4e38:       sub    $0x1,%edx
 0.00 :        4e3b:       cmp    %edx,%eax
 0.00 :        4e3d:       sete   %al
 0.00 :        4e40:       subl   $0x1,0xa4(%rbx)
 0.00 :        4e47:       addl   $0x1,0x9c(%rbx)
 0.00 :        4e4e:       movzbl %al,%eax
 0.00 :        4e51:       test   %eax,%eax
 0.00 :        4e53:       jne    4fe4 <deflate+0x9a4>
 0.00 :        4e59:       mov    0xa4(%rbx),%eax
 0.00 :        4e5f:       cmp    $0x102,%eax
 0.00 :        4e64:       ja     4dc8 <deflate+0x788>
 0.00 :        4e6a:       mov    %rbx,%rdi
 0.00 :        4e6d:       callq  3040 <crc32_combine64+0x260>
 0.00 :        4e72:       mov    0xa4(%rbx),%eax
 0.00 :        4e78:       cmp    $0x102,%eax
 0.00 :        4e7d:       ja     4dc8 <deflate+0x788>
 0.00 :        4e83:       test   %r12d,%r12d
 0.00 :        4e86:       je     487e <deflate+0x23e>
 0.00 :        4e8c:       test   %eax,%eax
 0.00 :        4e8e:       je     4bbb <deflate+0x57b>
 0.00 :        4e94:       cmp    $0x2,%eax
 0.00 :        4e97:       movl   $0x0,0x90(%rbx)
 0.00 :        4ea1:       ja     4dd2 <deflate+0x792>
 0.00 :        4ea7:       mov    0x50(%rbx),%rdx
 0.00 :        4eab:       jmpq   4df7 <deflate+0x7b7>
 0.00 :        4eb0:       movzbl 0x2(%rsi),%r8d
 0.00 :        4eb5:       cmp    %r8d,%edi
 0.00 :        4eb8:       jne    4df7 <deflate+0x7b7>
 0.00 :        4ebe:       movzbl 0x3(%rsi),%r8d
 0.00 :        4ec3:       cmp    %r8d,%edi
 0.00 :        4ec6:       jne    4df7 <deflate+0x7b7>
 0.00 :        4ecc:       add    $0x3,%rsi
 0.00 :        4ed0:       lea    0x102(%rdx,%r9,1),%r8
 0.00 :        4ed8:       jmp    4f38 <deflate+0x8f8>
 0.00 :        4eda:       nopw   0x0(%rax,%rax,1)
 0.00 :        4ee0:       movzbl 0x2(%rsi),%ecx
 0.00 :        4ee4:       cmp    %ecx,%edi
 0.00 :        4ee6:       jne    558b <deflate+0xf4b>
 0.00 :        4eec:       movzbl 0x3(%rsi),%ecx
 0.00 :        4ef0:       cmp    %ecx,%edi
 0.00 :        4ef2:       jne    5582 <deflate+0xf42>
 0.00 :        4ef8:       movzbl 0x4(%rsi),%ecx
 0.00 :        4efc:       cmp    %ecx,%edi
 0.00 :        4efe:       jne    5579 <deflate+0xf39>
 0.00 :        4f04:       movzbl 0x5(%rsi),%ecx
 0.00 :        4f08:       cmp    %ecx,%edi
 0.00 :        4f0a:       jne    5570 <deflate+0xf30>
 0.00 :        4f10:       movzbl 0x6(%rsi),%ecx
```

```
 0.00 :         4f14:       cmp    %ecx,%edi
 0.00 :         4f16:       jne    5567 <deflate+0xf27>
 0.00 :         4f1c:       movzbl 0x7(%rsi),%ecx
 0.00 :         4f20:       cmp    %ecx,%edi
 0.00 :         4f22:       jne    555e <deflate+0xf1e>
 0.00 :         4f28:       add    $0x8,%rsi
 0.00 :         4f2c:       movzbl (%rsi),%ecx
 0.00 :         4f2f:       cmp    %ecx,%edi
 0.00 :         4f31:       jne    4f44 <deflate+0x904>
 0.00 :         4f33:       cmp    %r8,%rsi
 0.00 :         4f36:       jae    4f44 <deflate+0x904>
 0.00 :         4f38:       movzbl 0x1(%rsi),%ecx
 0.00 :         4f3c:       cmp    %ecx,%edi
 0.00 :         4f3e:       je     4ee0 <deflate+0x8a0>
 0.00 :         4f40:       add    $0x1,%rsi
 0.00 :         4f44:       sub    %rsi,%r8
 0.00 :         4f47:       mov    %r13d,%ecx
 0.00 :         4f4a:       sub    %r8d,%ecx
 0.00 :         4f4d:       cmp    %eax,%ecx
 0.00 :         4f4f:       mov    %ecx,0x90(%rbx)
 0.00 :         4f55:       jbe    522e <deflate+0xbee>
 0.00 :         4f5b:       mov    %eax,0x90(%rbx)
 0.00 :         4f61:       mov    %eax,%ecx
 0.00 :         4f63:       mov    0x16fc(%rbx),%eax
 0.00 :         4f69:       mov    0x1700(%rbx),%rsi
 0.00 :         4f70:       lea    -0x3(%rcx),%edx
 0.00 :         4f73:       mov    %eax,%ecx
 0.00 :         4f75:       add    $0x1,%eax
 0.00 :         4f78:       movw   $0x1,(%rsi,%rcx,2)
 0.00 :         4f7e:       mov    0x16f0(%rbx),%rsi
 0.00 :         4f85:       mov    %dl,(%rsi,%rcx,1)
 0.00 :         4f88:       lea    0xcdb1(%rip),%rcx        # 11d40 <gzclose_w+0x36c0>
 0.00 :         4f8f:       movzbl %dl,%edx
 0.00 :         4f92:       mov    %eax,0x16fc(%rbx)
 0.00 :         4f98:       movzbl (%rcx,%rdx,1),%edx
 0.00 :         4f9c:       addw   $0x1,0x4c8(%rbx,%rdx,4)
 0.00 :         4fa5:       addw   $0x1,0x9b8(%rbx,%r14,4)
 0.00 :         4faf:       mov    0x16f8(%rbx),%edx
 0.00 :         4fb5:       sub    $0x1,%edx
 0.00 :         4fb8:       cmp    %edx,%eax
 0.00 :         4fba:       mov    0x90(%rbx),%edx
 0.00 :         4fc0:       movl   $0x0,0x90(%rbx)
 0.00 :         4fca:       sete   %al
 0.00 :         4fcd:       sub    %edx,0xa4(%rbx)
 0.00 :         4fd3:       add    %edx,0x9c(%rbx)
 0.00 :         4fd9:       movzbl %al,%eax
 0.00 :         4fdc:       test   %eax,%eax
 0.00 :         4fde:       je     4e59 <deflate+0x819>
 0.00 :         4fe4:       mov    0x9c(%rbx),%edx
 0.00 :         4fea:       mov    0x88(%rbx),%rax
 0.00 :         4ff1:       xor    %esi,%esi
 0.00 :         4ff3:       sub    %rax,%rdx
 0.00 :         4ff6:       test   %rax,%rax
 0.00 :         4ff9:       js     5001 <deflate+0x9c1>
 0.00 :         4ffb:       mov    %eax,%esi
 0.00 :         4ffd:       add    0x50(%rbx),%rsi
 0.00 :         5001:       xor    %ecx,%ecx
 0.00 :         5003:       mov    %rbx,%rdi
 0.00 :         5006:       callq  be40 <inflateMark+0x1be0>
 0.00 :         500b:       mov    0x9c(%rbx),%eax
 0.00 :         5011:       mov    (%rbx),%rdi
 0.00 :         5014:       mov    %rax,0x88(%rbx)
 0.00 :         501b:       callq  3380 <crc32_combine64+0x5a0>
 0.00 :         5020:       mov    (%rbx),%rax
```

```
 0.00 :         5023:       mov    0x20(%rax),%eax
 0.00 :         5026:       test   %eax,%eax
 0.00 :         5028:       jne    4e59 <deflate+0x819>
 0.00 :         502e:       jmpq   487e <deflate+0x23e>
 0.00 :         5033:       nopl   0x0(%rax,%rax,1)
 0.00 :         5038:       xor    %r14d,%r14d
 0.00 :         503b:       mov    0x30(%rbx),%rax
 0.00 :         503f:       mov    0x44(%rax),%edi
 0.00 :         5042:       test   %edi,%edi
 0.00 :         5044:       je     504f <deflate+0xa0f>
 0.00 :         5046:       cmp    %edx,%r8d
 0.00 :         5049:       jb     54ac <deflate+0xe6c>
 0.00 :         504f:       test   %r14d,%r14d
 0.00 :         5052:       je     50a4 <deflate+0xa64>
 0.00 :         5054:       mov    0x8(%rbx),%eax
 0.00 :         5057:       cmp    $0x5b,%eax
 0.00 :         505a:       jne    46cf <deflate+0x8f>
 0.00 :         5060:       mov    0x30(%rbx),%rax
 0.00 :         5064:       jmp    50b6 <deflate+0xa76>
 0.00 :         5066:       nopw   %cs:0x0(%rax,%rax,1)
 0.00 :         5070:       mov    0x8(%rdi),%r10d
 0.00 :         5074:       test   %r10d,%r10d
 0.00 :         5077:       je     4682 <deflate+0x42>
 0.00 :         507d:       jmpq   47ea <deflate+0x1aa>
 0.00 :         5082:       nopw   0x0(%rax,%rax,1)
 0.00 :         5088:       mov    %r8d,%esi
 0.00 :         508b:       add    0x10(%rbx),%rsi
 0.00 :         508f:       mov    0x60(%rbp),%rdi
 0.00 :         5093:       sub    %r8d,%edx
 0.00 :         5096:       callq  2080 <crc32@plt>
 0.00 :         509b:       mov    %rax,0x60(%rbp)
 0.00 :         509f:       jmpq   4a9f <deflate+0x45f>
 0.00 :         50a4:       mov    0x30(%rbx),%rax
 0.00 :         50a8:       movl   $0x0,0x38(%rbx)
 0.00 :         50af:       movl   $0x5b,0x8(%rbx)
 0.00 :         50b6:       cmpq   $0x0,0x38(%rax)
 0.00 :         50bb:       je     5258 <deflate+0xc18>
 0.00 :         50c1:       mov    %edx,%r8d
 0.00 :         50c4:       jmp    50f8 <deflate+0xab8>
 0.00 :         50c6:       nopw   %cs:0x0(%rax,%rax,1)
 0.00 :         50d0:       mov    0x38(%rbx),%esi
 0.00 :         50d3:       mov    0x38(%rax),%rax
 0.00 :         50d7:       add    $0x1,%edx
 0.00 :         50da:       mov    %esi,%edi
 0.00 :         50dc:       add    $0x1,%esi
 0.00 :         50df:       movzbl (%rax,%rdi,1),%eax
 0.00 :         50e3:       mov    %esi,0x38(%rbx)
 0.00 :         50e6:       mov    0x10(%rbx),%rsi
 0.00 :         50ea:       test   %al,%al
 0.00 :         50ec:       mov    %al,(%rsi,%rcx,1)
 0.00 :         50ef:       mov    %edx,0x28(%rbx)
 0.00 :         50f2:       je     5138 <deflate+0xaf8>
 0.00 :         50f4:       mov    0x30(%rbx),%rax
 0.00 :         50f8:       mov    %edx,%ecx
 0.00 :         50fa:       cmp    0x18(%rbx),%rcx
 0.00 :         50fe:       jne    50d0 <deflate+0xa90>
 0.00 :         5100:       mov    0x44(%rax),%esi
 0.00 :         5103:       test   %esi,%esi
 0.00 :         5105:       je     5110 <deflate+0xad0>
 0.00 :         5107:       cmp    %edx,%r8d
 0.00 :         510a:       jb     51ec <deflate+0xbac>
 0.00 :         5110:       mov    %rbp,%rdi
 0.00 :         5113:       callq  3380 <crc32_combine64+0x5a0>
 0.00 :         5118:       mov    0x28(%rbx),%r8d
```

```
 0.00 :        511c:       mov    %r8d,%ecx
 0.00 :        511f:       cmp    0x18(%rbx),%rcx
 0.00 :        5123:       mov    %r8d,%edx
 0.00 :        5126:       je     5522 <deflate+0xee2>
 0.00 :        512c:       mov    0x30(%rbx),%rax
 0.00 :        5130:       jmp    50d0 <deflate+0xa90>
 0.00 :        5132:       nopw   0x0(%rax,%rax,1)
 0.00 :        5138:       xor    %r14d,%r14d
 0.00 :        513b:       mov    0x30(%rbx),%rax
 0.00 :        513f:       mov    0x44(%rax),%ecx
 0.00 :        5142:       test   %ecx,%ecx
 0.00 :        5144:       je     514f <deflate+0xb0f>
 0.00 :        5146:       cmp    %edx,%r8d
 0.00 :        5149:       jb     54cb <deflate+0xe8b>
 0.00 :        514f:       test   %r14d,%r14d
 0.00 :        5152:       je     51cc <deflate+0xb8c>
 0.00 :        5154:       mov    0x8(%rbx),%eax
 0.00 :        5157:       cmp    $0x67,%eax
 0.00 :        515a:       jne    46d8 <deflate+0x98>
 0.00 :        5160:       mov    0x30(%rbx),%rax
 0.00 :        5164:       mov    0x44(%rax),%eax
 0.00 :        5167:       test   %eax,%eax
 0.00 :        5169:       je     51c0 <deflate+0xb80>
 0.00 :        516b:       lea    0x2(%rdx),%eax
 0.00 :        516e:       mov    %eax,%ecx
 0.00 :        5170:       cmp    0x18(%rbx),%rcx
 0.00 :        5174:       ja     5270 <deflate+0xc30>
 0.00 :        517a:       mov    0x60(%rbp),%rdi
 0.00 :        517e:       mov    0x10(%rbx),%rsi
 0.00 :        5182:       mov    %edx,%ecx
 0.00 :        5184:       mov    %dil,(%rsi,%rcx,1)
 0.00 :        5188:       lea    0x1(%rdx),%ecx
 0.00 :        518b:       mov    0x60(%rbp),%rdx
 0.00 :        518f:       mov    0x10(%rbx),%rsi
 0.00 :        5193:       xor    %edi,%edi
 0.00 :        5195:       shr    $0x8,%rdx
 0.00 :        5199:       mov    %dl,(%rsi,%rcx,1)
 0.00 :        519c:       xor    %edx,%edx
 0.00 :        519e:       mov    %eax,0x28(%rbx)
 0.00 :        51a1:       xor    %esi,%esi
 0.00 :        51a3:       callq  2080 <crc32@plt>
 0.00 :        51a8:       mov    0x28(%rbx),%edx
 0.00 :        51ab:       mov    %rax,0x60(%rbp)
 0.00 :        51af:       movl   $0x71,0x8(%rbx)
 0.00 :        51b6:       jmpq   46d8 <deflate+0x98>
 0.00 :        51bb:       nopl   0x0(%rax,%rax,1)
 0.00 :        51c0:       movl   $0x71,0x8(%rbx)
 0.00 :        51c7:       jmpq   46d8 <deflate+0x98>
 0.00 :        51cc:       movl   $0x67,0x8(%rbx)
 0.00 :        51d3:       mov    0x30(%rbx),%rax
 0.00 :        51d7:       jmp    5164 <deflate+0xb24>
 0.00 :        51d9:       movl   $0x0,0x38(%rbx)
 0.00 :        51e0:       movl   $0x49,0x8(%rbx)
 0.00 :        51e7:       jmpq   4ccc <deflate+0x68c>
 0.00 :        51ec:       mov    %r8d,%esi
 0.00 :        51ef:       add    0x10(%rbx),%rsi
 0.00 :        51f3:       mov    0x60(%rbp),%rdi
 0.00 :        51f7:       sub    %r8d,%edx
 0.00 :        51fa:       callq  2080 <crc32@plt>
 0.00 :        51ff:       mov    %rax,0x60(%rbp)
 0.00 :        5203:       jmpq   5110 <deflate+0xad0>
 0.00 :        5208:       mov    %r8d,%esi
 0.00 :        520b:       add    0x10(%rbx),%rsi
 0.00 :        520f:       mov    0x60(%rbp),%rdi
```

```
 0.00 :         5213:       sub    %r8d,%edx
 0.00 :         5216:       callq  2080 <crc32@plt>
 0.00 :         521b:       mov    %rax,0x60(%rbp)
 0.00 :         521f:       jmpq   4d26 <deflate+0x6e6>
 0.00 :         5224:       mov    $0xfffffffe,%eax
 0.00 :         5229:       jmpq   488d <deflate+0x24d>
 0.00 :         522e:       cmp    $0x2,%ecx
 0.00 :         5231:       ja     4f63 <deflate+0x923>
 0.00 :         5237:       jmpq   4df7 <deflate+0x7b7>
 0.00 :         523c:       nopl   0x0(%rax)
 0.00 :         5240:       mov    0x30(%rbx),%rax
 0.00 :         5244:       jmpq   4c98 <deflate+0x658>
 0.00 :         5249:       movl   $0x49,0x8(%rbx)
 0.00 :         5250:       mov    %r8d,%edx
 0.00 :         5253:       jmpq   4ccc <deflate+0x68c>
 0.00 :         5258:       movl   $0x67,0x8(%rbx)
 0.00 :         525f:       jmpq   5164 <deflate+0xb24>
 0.00 :         5264:       movl   $0x5b,0x8(%rbx)
 0.00 :         526b:       jmpq   50b6 <deflate+0xa76>
 0.00 :         5270:       mov    %rbp,%rdi
 0.00 :         5273:       callq  3380 <crc32_combine64+0x5a0>
 0.00 :         5278:       mov    0x28(%rbx),%edx
 0.00 :         527b:       lea    0x2(%rdx),%eax
 0.00 :         527e:       mov    %eax,%ecx
 0.00 :         5280:       cmp    0x18(%rbx),%rcx
 0.00 :         5284:       ja     46d8 <deflate+0x98>
 0.00 :         528a:       jmpq   517a <deflate+0xb3a>
 0.00 :         528f:       nop
 0.00 :         5290:       mov    0x28(%rbx),%eax
 0.00 :         5293:       mov    0x60(%rbp),%rsi
 0.00 :         5297:       mov    0x10(%rbx),%rcx
 0.00 :         529b:       mov    %eax,%edx
 0.00 :         529d:       mov    %sil,(%rcx,%rdx,1)
 0.00 :         52a1:       mov    0x60(%rbp),%rdx
 0.00 :         52a5:       lea    0x1(%rax),%ecx
 0.00 :         52a8:       mov    0x10(%rbx),%rsi
 0.00 :         52ac:       shr    $0x8,%rdx
 0.00 :         52b0:       mov    %dl,(%rsi,%rcx,1)
 0.00 :         52b3:       mov    0x60(%rbp),%rdx
 0.00 :         52b7:       lea    0x2(%rax),%ecx
 0.00 :         52ba:       mov    0x10(%rbx),%rsi
 0.00 :         52be:       shr    $0x10,%rdx
 0.00 :         52c2:       mov    %dl,(%rsi,%rcx,1)
 0.00 :         52c5:       mov    0x60(%rbp),%rdx
 0.00 :         52c9:       lea    0x3(%rax),%ecx
 0.00 :         52cc:       mov    0x10(%rbx),%rsi
 0.00 :         52d0:       shr    $0x18,%rdx
 0.00 :         52d4:       mov    %dl,(%rsi,%rcx,1)
 0.00 :         52d7:       mov    0x10(%rbp),%rsi
 0.00 :         52db:       lea    0x4(%rax),%edx
 0.00 :         52de:       mov    0x10(%rbx),%rcx
 0.00 :         52e2:       mov    %sil,(%rcx,%rdx,1)
 0.00 :         52e6:       mov    0x10(%rbp),%rdx
 0.00 :         52ea:       lea    0x5(%rax),%ecx
 0.00 :         52ed:       mov    0x10(%rbx),%rsi
 0.00 :         52f1:       shr    $0x8,%rdx
 0.00 :         52f5:       mov    %dl,(%rsi,%rcx,1)
 0.00 :         52f8:       mov    0x10(%rbp),%rdx
 0.00 :         52fc:       lea    0x6(%rax),%ecx
 0.00 :         52ff:       mov    0x10(%rbx),%rsi
 0.00 :         5303:       shr    $0x10,%rdx
 0.00 :         5307:       mov    %dl,(%rsi,%rcx,1)
 0.00 :         530a:       mov    0x10(%rbp),%rdx
 0.00 :         530e:       lea    0x7(%rax),%ecx
```

```
 0.00 :        5311:       mov    0x10(%rbx),%rsi
 0.00 :        5315:       add    $0x8,%eax
 0.00 :        5318:       shr    $0x18,%rdx
 0.00 :        531c:       mov    %dl,(%rsi,%rcx,1)
 0.00 :        531f:       mov    %eax,0x28(%rbx)
 0.00 :        5322:       jmpq   47b6 <deflate+0x176>
 0.00 :        5327:       xor    %edx,%edx
 0.00 :        5329:       xor    %esi,%esi
 0.00 :        532b:       xor    %edi,%edi
 0.00 :        532d:       callq  2080 <crc32@plt>
 0.00 :        5332:       mov    0x28(%rbx),%edx
 0.00 :        5335:       mov    0x10(%rbx),%rcx
 0.00 :        5339:       mov    %rax,0x60(%rbp)
 0.00 :        533d:       mov    %edx,%eax
 0.00 :        533f:       movb   $0x1f,(%rcx,%rax,1)
 0.00 :        5343:       mov    0x10(%rbx),%rcx
 0.00 :        5347:       lea    0x1(%rdx),%eax
 0.00 :        534a:       movb   $0x8b,(%rcx,%rax,1)
 0.00 :        534e:       mov    0x10(%rbx),%rcx
 0.00 :        5352:       lea    0x2(%rdx),%eax
 0.00 :        5355:       movb   $0x8,(%rcx,%rax,1)
 0.00 :        5359:       mov    0x30(%rbx),%rax
 0.00 :        535d:       lea    0x3(%rdx),%ecx
 0.00 :        5360:       mov    %ecx,0x28(%rbx)
 0.00 :        5363:       test   %rax,%rax
 0.00 :        5366:       je     55a5 <deflate+0xf65>
 0.00 :        536c:       mov    (%rax),%r8d
 0.00 :        536f:       mov    %ecx,%esi
 0.00 :        5371:       add    0x10(%rbx),%rsi
 0.00 :        5375:       test   %r8d,%r8d
 0.00 :        5378:       setne  %dil
 0.00 :        537c:       cmpl   $0x1,0x44(%rax)
 0.00 :        5380:       sbb    %ecx,%ecx
 0.00 :        5382:       not    %ecx
 0.00 :        5384:       and    $0x2,%ecx
 0.00 :        5387:       add    %ecx,%edi
 0.00 :        5389:       cmpq   $0x1,0x18(%rax)
 0.00 :        538e:       sbb    %ecx,%ecx
 0.00 :        5390:       not    %ecx
 0.00 :        5392:       and    $0x4,%ecx
 0.00 :        5395:       add    %ecx,%edi
 0.00 :        5397:       cmpq   $0x1,0x28(%rax)
 0.00 :        539c:       sbb    %ecx,%ecx
 0.00 :        539e:       not    %ecx
 0.00 :        53a0:       and    $0x8,%ecx
 0.00 :        53a3:       add    %edi,%ecx
 0.00 :        53a5:       cmpq   $0x1,0x38(%rax)
 0.00 :        53aa:       sbb    %eax,%eax
 0.00 :        53ac:       not    %eax
 0.00 :        53ae:       and    $0x10,%eax
 0.00 :        53b1:       add    %ecx,%eax
 0.00 :        53b3:       mov    %al,(%rsi)
 0.00 :        53b5:       mov    0x30(%rbx),%rsi
 0.00 :        53b9:       lea    0x4(%rdx),%eax
 0.00 :        53bc:       mov    0x10(%rbx),%rcx
 0.00 :        53c0:       mov    0x8(%rsi),%rsi
 0.00 :        53c4:       mov    %sil,(%rcx,%rax,1)
 0.00 :        53c8:       mov    0x30(%rbx),%rax
 0.00 :        53cc:       lea    0x5(%rdx),%ecx
 0.00 :        53cf:       mov    0x10(%rbx),%rsi
 0.00 :        53d3:       mov    0x8(%rax),%rax
 0.00 :        53d7:       shr    $0x8,%rax
 0.00 :        53db:       mov    %al,(%rsi,%rcx,1)
 0.00 :        53de:       mov    0x30(%rbx),%rax
```

```
 0.00 :        53e2:       lea    0x6(%rdx),%ecx
 0.00 :        53e5:       mov    0x10(%rbx),%rsi
 0.00 :        53e9:       mov    0x8(%rax),%rax
 0.00 :        53ed:       shr    $0x10,%rax
 0.00 :        53f1:       mov    %al,(%rsi,%rcx,1)
 0.00 :        53f4:       mov    0x30(%rbx),%rax
 0.00 :        53f8:       lea    0x7(%rdx),%ecx
 0.00 :        53fb:       mov    0x10(%rbx),%rsi
 0.00 :        53ff:       mov    0x8(%rax),%rax
 0.00 :        5403:       shr    $0x18,%rax
 0.00 :        5407:       mov    %al,(%rsi,%rcx,1)
 0.00 :        540a:       mov    0xb4(%rbx),%esi
 0.00 :        5410:       lea    0x8(%rdx),%eax
 0.00 :        5413:       mov    $0x2,%ecx
 0.00 :        5418:       mov    %eax,0x28(%rbx)
 0.00 :        541b:       add    0x10(%rbx),%rax
 0.00 :        541f:       cmp    $0x9,%esi
 0.00 :        5422:       je     5436 <deflate+0xdf6>
 0.00 :        5424:       cmpl   $0x1,0xb8(%rbx)
 0.00 :        542b:       jle    5594 <deflate+0xf54>
 0.00 :        5431:       mov    $0x4,%ecx
 0.00 :        5436:       mov    %cl,(%rax)
 0.00 :        5438:       mov    0x30(%rbx),%rsi
 0.00 :        543c:       lea    0x9(%rdx),%eax
 0.00 :        543f:       mov    0x10(%rbx),%rcx
 0.00 :        5443:       lea    0xa(%rdx),%r8d
 0.00 :        5447:       mov    0x14(%rsi),%esi
 0.00 :        544a:       mov    %sil,(%rcx,%rax,1)
 0.00 :        544e:       mov    0x30(%rbx),%rsi
 0.00 :        5452:       mov    %r8d,0x28(%rbx)
 0.00 :        5456:       cmpq   $0x0,0x18(%rsi)
 0.00 :        545b:       mov    %rsi,%rax
 0.00 :        545e:       je     548e <deflate+0xe4e>
 0.00 :        5460:       mov    0x20(%rsi),%ecx
 0.00 :        5463:       mov    0x10(%rbx),%rax
 0.00 :        5467:       mov    %cl,(%rax,%r8,1)
 0.00 :        546b:       mov    0x30(%rbx),%rax
 0.00 :        546f:       lea    0xb(%rdx),%ecx
 0.00 :        5472:       mov    0x10(%rbx),%rsi
 0.00 :        5476:       lea    0xc(%rdx),%r8d
 0.00 :        547a:       mov    0x20(%rax),%eax
 0.00 :        547d:       shr    $0x8,%eax
 0.00 :        5480:       mov    %al,(%rsi,%rcx,1)
 0.00 :        5483:       mov    0x30(%rbx),%rsi
 0.00 :        5487:       mov    %r8d,0x28(%rbx)
 0.00 :        548b:       mov    %rsi,%rax
 0.00 :        548e:       mov    0x44(%rsi),%edi
 0.00 :        5491:       test   %edi,%edi
 0.00 :        5493:       jne    553a <deflate+0xefa>
 0.00 :        5499:       movl   $0x0,0x38(%rbx)
 0.00 :        54a0:       movl   $0x45,0x8(%rbx)
 0.00 :        54a7:       jmpq   4a31 <deflate+0x3f1>
 0.00 :        54ac:       mov    %r8d,%esi
 0.00 :        54af:       add    0x10(%rbx),%rsi
 0.00 :        54b3:       mov    0x60(%rbp),%rdi
 0.00 :        54b7:       sub    %r8d,%edx
 0.00 :        54ba:       callq  2080 <crc32@plt>
 0.00 :        54bf:       mov    0x28(%rbx),%edx
 0.00 :        54c2:       mov    %rax,0x60(%rbp)
 0.00 :        54c6:       jmpq   504f <deflate+0xa0f>
 0.00 :        54cb:       mov    %r8d,%esi
 0.00 :        54ce:       add    0x10(%rbx),%rsi
 0.00 :        54d2:       mov    0x60(%rbp),%rdi
 0.00 :        54d6:       sub    %r8d,%edx
```

```
 0.00 :         54d9:       callq  2080 <crc32@plt>
 0.00 :         54de:       mov    0x28(%rbx),%edx
 0.00 :         54e1:       mov    %rax,0x60(%rbp)
 0.00 :         54e5:       jmpq   514f <deflate+0xb0f>
 0.00 :         54ea:       mov    %r8d,%esi
 0.00 :         54ed:       add    0x10(%rbx),%rsi
 0.00 :         54f1:       mov    0x60(%rbp),%rdi
 0.00 :         54f5:       sub    %r8d,%edx
 0.00 :         54f8:       callq  2080 <crc32@plt>
 0.00 :         54fd:       mov    0x28(%rbx),%edx
 0.00 :         5500:       mov    %rax,0x60(%rbp)
 0.00 :         5504:       mov    0x30(%rbx),%rax
 0.00 :         5508:       jmpq   4caa <deflate+0x66a>
 0.00 :         550d:       mov    $0x1,%eax
 0.00 :         5512:       jmpq   488d <deflate+0x24d>
 0.00 :         5517:       mov    $0x1,%r14d
 0.00 :         551d:       jmpq   503b <deflate+0x9fb>
 0.00 :         5522:       mov    $0x1,%r14d
 0.00 :         5528:       jmpq   513b <deflate+0xafb>
 0.00 :         552d:       mov    %rbx,%rdi
 0.00 :         5530:       callq  bd60 <inflateMark+0x1b00>
 0.00 :         5535:       jmpq   4913 <deflate+0x2d3>
 0.00 :         553a:       mov    0x10(%rbx),%rsi
 0.00 :         553e:       mov    0x60(%rbp),%rdi
 0.00 :         5542:       mov    %r8d,%edx
 0.00 :         5545:       callq  2080 <crc32@plt>
 0.00 :         554a:       mov    0x30(%rbx),%rsi
 0.00 :         554e:       mov    %rax,0x60(%rbp)
 0.00 :         5552:       mov    0x28(%rbx),%r8d
 0.00 :         5556:       mov    %rsi,%rax
 0.00 :         5559:       jmpq   5499 <deflate+0xe59>
 0.00 :         555e:       add    $0x7,%rsi
 0.00 :         5562:       jmpq   4f44 <deflate+0x904>
 0.00 :         5567:       add    $0x6,%rsi
 0.00 :         556b:       jmpq   4f44 <deflate+0x904>
 0.00 :         5570:       add    $0x5,%rsi
 0.00 :         5574:       jmpq   4f44 <deflate+0x904>
 0.00 :         5579:       add    $0x4,%rsi
 0.00 :         557d:       jmpq   4f44 <deflate+0x904>
 0.00 :         5582:       add    $0x3,%rsi
 0.00 :         5586:       jmpq   4f44 <deflate+0x904>
 0.00 :         558b:       add    $0x2,%rsi
 0.00 :         558f:       jmpq   4f44 <deflate+0x904>
 0.00 :         5594:       cmp    $0x1,%esi
 0.00 :         5597:       jle    5431 <deflate+0xdf1>
 0.00 :         559d:       xor    %ecx,%ecx
 0.00 :         559f:       nop
 0.00 :         55a0:       jmpq   5436 <deflate+0xdf6>
 0.00 :         55a5:       mov    0x10(%rbx),%rax
 0.00 :         55a9:       movb   $0x0,(%rax,%rcx,1)
 0.00 :         55ad:       mov    0x10(%rbx),%rcx
 0.00 :         55b1:       lea    0x4(%rdx),%eax
 0.00 :         55b4:       movb   $0x0,(%rcx,%rax,1)
 0.00 :         55b8:       mov    0x10(%rbx),%rcx
 0.00 :         55bc:       lea    0x5(%rdx),%eax
 0.00 :         55bf:       movb   $0x0,(%rcx,%rax,1)
 0.00 :         55c3:       mov    0x10(%rbx),%rcx
 0.00 :         55c7:       lea    0x6(%rdx),%eax
 0.00 :         55ca:       movb   $0x0,(%rcx,%rax,1)
 0.00 :         55ce:       mov    0x10(%rbx),%rcx
 0.00 :         55d2:       lea    0x7(%rdx),%eax
 0.00 :         55d5:       movb   $0x0,(%rcx,%rax,1)
 0.00 :         55d9:       mov    0xb4(%rbx),%esi
 0.00 :         55df:       lea    0x8(%rdx),%eax
```

```
    0.00 :          55e2:       mov    $0x2,%ecx
    0.00 :          55e7:       mov    %eax,0x28(%rbx)
    0.00 :          55ea:       add    0x10(%rbx),%rax
    0.00 :          55ee:       cmp    $0x9,%esi
    0.00 :          55f1:       je     5601 <deflate+0xfc1>
    0.00 :          55f3:       cmpl   $0x1,0xb8(%rbx)
    0.00 :          55fa:       jle    5620 <deflate+0xfe0>
    0.00 :          55fc:       mov    $0x4,%ecx
    0.00 :          5601:       mov    %cl,(%rax)
    0.00 :          5603:       mov    0x10(%rbx),%rcx
    0.00 :          5607:       lea    0x9(%rdx),%eax
    0.00 :          560a:       add    $0xa,%edx
    0.00 :          560d:       movb   $0x3,(%rcx,%rax,1)
    0.00 :          5611:       mov    %edx,0x28(%rbx)
    0.00 :          5614:       movl   $0x71,0x8(%rbx)
    0.00 :          561b:       jmpq   46d8 <deflate+0x98>
    0.00 :          5620:       cmp    $0x1,%esi
    0.00 :          5623:       jle    55fc <deflate+0xfbc>
    0.00 :          5625:       xor    %ecx,%ecx
    0.00 :          5627:       jmp    5601 <deflate+0xfc1>
 Percent |       Source code & Disassembly of libpng15.so.15.13.0
------------------------------------------------
         :
         :
         :
         :       Disassembly of section .text:
         :
         :       00000000000064a0 <png_get_io_ptr>:
  100.00 :          64a0:       test   %rdi,%rdi
    0.00 :          64a3:       je     64b0 <png_get_io_ptr+0x10>
    0.00 :          64a5:       mov    0xf8(%rdi),%rax
    0.00 :          64ac:       retq
    0.00 :          64ad:       nopl   (%rax)
    0.00 :          64b0:       xor    %eax,%eax
    0.00 :          64b2:       retq
 Percent |       Source code & Disassembly of libpng15.so.15.13.0
------------------------------------------------
         :
         :
         :
         :       Disassembly of section .text:
         :
         :       000000000001cc00 <png_write_row>:
    0.00 :          1cc00:       push   %rbp
    0.00 :          1cc01:       push   %rbx
    0.00 :          1cc02:       mov    %rdi,%rbx
    0.00 :          1cc05:       sub    $0x38,%rsp
    0.00 :          1cc09:       test   %rdi,%rdi
    0.00 :          1cc0c:       je     1cd6b <png_write_row+0x16b>
    0.00 :          1cc12:       mov    0x1ec(%rdi),%r9d
    0.00 :          1cc19:       test   %r9d,%r9d
    0.00 :          1cc1c:       jne    1cc43 <png_write_row+0x43>
    0.00 :          1cc1e:       cmpb   $0x0,0x24d(%rdi)
    0.00 :          1cc25:       jne    1cc43 <png_write_row+0x43>
    0.00 :          1cc27:       testb  $0x4,0x11d(%rdi)
    0.00 :          1cc2e:       je     1cf39 <png_write_row+0x339>
    0.00 :          1cc34:       mov    %rsi,0x8(%rsp)
    0.00 :          1cc39:       callq  214d0 <png_write_chunk+0x20a0>
    0.00 :          1cc3e:       mov    0x8(%rsp),%rsi
    0.00 :          1cc43:       cmpb   $0x0,0x24c(%rbx)
    0.00 :          1cc4a:       je     1cc68 <png_write_row+0x68>
    0.00 :          1cc4c:       testb  $0x2,0x124(%rbx)
    0.00 :          1cc53:       je     1cc68 <png_write_row+0x68>
    0.00 :          1cc55:       cmpb   $0x6,0x24d(%rbx)
```

```
  0.00 :        1cc5c:       jbe    1ce10 <png_write_row+0x210>
  0.00 :        1cc62:       nopw   0x0(%rax,%rax,1)
  0.00 :        1cc68:       movzbl 0x24f(%rbx),%eax
  0.00 :        1cc6f:       movzbl 0x254(%rbx),%edx
  0.00 :        1cc76:       mov    0x1dc(%rbx),%ecx
  0.00 :        1cc7c:       mov    %al,0x20(%rsp)
  0.00 :        1cc80:       movzbl 0x251(%rbx),%eax
  0.00 :        1cc87:       mov    %ecx,0x10(%rsp)
  0.00 :        1cc8b:       mov    %dl,0x22(%rsp)
  0.00 :        1cc8f:       mov    %al,0x21(%rsp)
  0.00 :        1cc93:       imul   %edx,%eax
  0.00 :        1cc96:       cmp    $0x7,%al
  0.00 :        1cc98:       mov    %al,0x23(%rsp)
  0.00 :        1cc9c:       ja     1cd78 <png_write_row+0x178>
  0.00 :        1cca2:       movzbl %al,%edx
  0.00 :        1cca5:       imul   %rcx,%rdx
  0.00 :        1cca9:       add    $0x7,%rdx
  0.00 :        1ccad:       shr    $0x3,%rdx
  0.00 :        1ccb1:       mov    0x200(%rbx),%rdi
  0.00 :        1ccb8:       mov    %rdx,0x18(%rsp)
  0.00 :        1ccbd:       add    $0x1,%rdi
  0.00 :        1ccc1:       callq  5560 <memcpy@plt>
  0.00 :        1ccc6:       cmpb   $0x0,0x24c(%rbx)
  0.00 :        1cccd:       je     1cdc0 <png_write_row+0x1c0>
  0.00 :        1ccd3:       movzbl 0x24d(%rbx),%edx
  0.00 :        1ccda:       mov    0x124(%rbx),%eax
  0.00 :        1cce0:       cmp    $0x5,%dl
  0.00 :        1cce3:       jbe    1cd90 <png_write_row+0x190>
  0.00 :        1cce9:       test   %eax,%eax
  0.00 :        1cceb:       jne    1cdce <png_write_row+0x1ce>
  0.00 :        1ccf1:       movzbl 0x23(%rsp),%eax
  0.00 :        1ccf6:       cmp    0x252(%rbx),%al
  0.00 :        1ccfc:       jne    1cf2a <png_write_row+0x32a>
  0.00 :        1cd02:       cmp    0x257(%rbx),%al
  0.00 :        1cd08:       jne    1cf2a <png_write_row+0x32a>
  0.00 :        1cd0e:       movabs $0xff00000004,%rax
  0.00 :        1cd18:       and    0x408(%rbx),%rax
  0.00 :        1cd1f:       movabs $0x4000000004,%rdx
  0.00 :        1cd29:       lea    0x10(%rsp),%rbp
  0.00 :        1cd2e:       cmp    %rdx,%rax
  0.00 :        1cd31:       je     1ce60 <png_write_row+0x260>
  0.00 :        1cd37:       cmpb   $0x3,0x20(%rsp)
  0.00 :        1cd3c:       je     1cde8 <png_write_row+0x1e8>
  0.00 :        1cd42:       mov    %rbp,%rsi
  0.00 :        1cd45:       mov    %rbx,%rdi
  0.00 :        1cd48:       callq  21bf0 <png_write_chunk+0x27c0>
  0.00 :        1cd4d:       mov    0x300(%rbx),%rax
100.00 :        1cd54:       test   %rax,%rax
  0.00 :        1cd57:       je     1cd6b <png_write_row+0x16b>
  0.00 :        1cd59:       movzbl 0x24d(%rbx),%edx
  0.00 :        1cd60:       mov    0x1ec(%rbx),%esi
  0.00 :        1cd66:       mov    %rbx,%rdi
  0.00 :        1cd69:       callq  *%rax
  0.00 :        1cd6b:       add    $0x38,%rsp
  0.00 :        1cd6f:       pop    %rbx
  0.00 :        1cd70:       pop    %rbp
  0.00 :        1cd71:       retq
  0.00 :        1cd72:       nopw   0x0(%rax,%rax,1)
  0.00 :        1cd78:       shr    $0x3,%al
  0.00 :        1cd7b:       movzbl %al,%edx
  0.00 :        1cd7e:       imul   %rcx,%rdx
  0.00 :        1cd82:       jmpq   1ccb1 <png_write_row+0xb1>
  0.00 :        1cd87:       nopw   0x0(%rax,%rax,1)
  0.00 :        1cd90:       test   $0x2,%al
```

```
 0.00 :        1cd92:       je     1cce9 <png_write_row+0xe9>
 0.00 :        1cd98:       mov    0x200(%rbx),%rsi
 0.00 :        1cd9f:       lea    0x10(%rsp),%rbp
 0.00 :        1cda4:       mov    %rbp,%rdi
 0.00 :        1cda7:       add    $0x1,%rsi
 0.00 :        1cdab:       callq  21910 <png_write_chunk+0x24e0>
 0.00 :        1cdb0:       mov    0x10(%rsp),%r8d
 0.00 :        1cdb5:       test   %r8d,%r8d
 0.00 :        1cdb8:       je     1ce48 <png_write_row+0x248>
 0.00 :        1cdbe:       xchg   %ax,%ax
 0.00 :        1cdc0:       mov    0x124(%rbx),%eax
 0.00 :        1cdc6:       test   %eax,%eax
 0.00 :        1cdc8:       je     1ccf1 <png_write_row+0xf1>
 0.00 :        1cdce:       lea    0x10(%rsp),%rbp
 0.00 :        1cdd3:       mov    %rbx,%rdi
 0.00 :        1cdd6:       mov    %rbp,%rsi
 0.00 :        1cdd9:       callq  1e590 <png_write_frame_tail+0x680>
 0.00 :        1cdde:       jmpq   1ccf1 <png_write_row+0xf1>
 0.00 :        1cde3:       nopl   0x0(%rax,%rax,1)
 0.00 :        1cde8:       mov    0x244(%rbx),%edi
 0.00 :        1cdee:       test   %edi,%edi
 0.00 :        1cdf0:       js     1cd42 <png_write_row+0x142>
 0.00 :        1cdf6:       mov    %rbp,%rsi
 0.00 :        1cdf9:       mov    %rbx,%rdi
 0.00 :        1cdfc:       callq  1bbe0 <png_set_invert_mono+0x490>
 0.00 :        1ce01:       jmpq   1cd42 <png_write_row+0x142>
 0.00 :        1ce06:       nopw   %cs:0x0(%rax,%rax,1)
 0.00 :        1ce10:       movzbl 0x24d(%rbx),%edx
 0.00 :        1ce17:       lea    0x90e2(%rip),%rax        # 25f00 <png_write_chunk+0x6ad0>
 0.00 :        1ce1e:       movslq (%rax,%rdx,4),%rdx
 0.00 :        1ce22:       add    %rdx,%rax
 0.00 :        1ce25:       jmpq   *%rax
 0.00 :        1ce27:       nopw   0x0(%rax,%rax,1)
 0.00 :        1ce30:       mov    0x1ec(%rbx),%eax
 0.00 :        1ce36:       and    $0x7,%eax
 0.00 :        1ce39:       cmp    $0x4,%eax
 0.00 :        1ce3c:       je     1cc68 <png_write_row+0x68>
 0.00 :        1ce42:       nopw   0x0(%rax,%rax,1)
 0.00 :        1ce48:       mov    %rbx,%rdi
 0.00 :        1ce4b:       callq  216e0 <png_write_chunk+0x22b0>
 0.00 :        1ce50:       add    $0x38,%rsp
 0.00 :        1ce54:       pop    %rbx
 0.00 :        1ce55:       pop    %rbp
 0.00 :        1ce56:       retq
 0.00 :        1ce57:       nopw   0x0(%rax,%rax,1)
 0.00 :        1ce60:       mov    0x200(%rbx),%rsi
 0.00 :        1ce67:       mov    %rbp,%rdi
 0.00 :        1ce6a:       add    $0x1,%rsi
 0.00 :        1ce6e:       callq  1e7b0 <png_write_frame_tail+0x8a0>
 0.00 :        1ce73:       jmpq   1cd37 <png_write_row+0x137>
 0.00 :        1ce78:       nopl   0x0(%rax,%rax,1)
 0.00 :        1ce80:       testb  $0x1,0x1ec(%rbx)
 0.00 :        1ce87:       jne    1ce48 <png_write_row+0x248>
 0.00 :        1ce89:       cmpl   $0x1,0x1d0(%rbx)
 0.00 :        1ce90:       ja     1cc68 <png_write_row+0x68>
 0.00 :        1ce96:       jmp    1ce48 <png_write_row+0x248>
 0.00 :        1ce98:       nopl   0x0(%rax,%rax,1)
 0.00 :        1cea0:       mov    0x1ec(%rbx),%eax
 0.00 :        1cea6:       and    $0x3,%eax
 0.00 :        1cea9:       cmp    $0x2,%eax
 0.00 :        1ceac:       jne    1ce48 <png_write_row+0x248>
 0.00 :        1ceae:       jmpq   1cc68 <png_write_row+0x68>
 0.00 :        1ceb3:       nopl   0x0(%rax,%rax,1)
 0.00 :        1ceb8:       testb  $0x3,0x1ec(%rbx)
```

```
    0.00 :          1cebf:       jne    1ce48 <png_write_row+0x248>
    0.00 :          1cec1:       cmpl   $0x2,0x1d0(%rbx)
    0.00 :          1cec8:       jbe    1ce48 <png_write_row+0x248>
    0.00 :          1cece:       jmpq   1cc68 <png_write_row+0x68>
    0.00 :          1ced3:       nopl   0x0(%rax,%rax,1)
    0.00 :          1ced8:       testb  $0x7,0x1ec(%rbx)
    0.00 :          1cedf:       jne    1ce48 <png_write_row+0x248>
    0.00 :          1cee5:       cmpl   $0x4,0x1d0(%rbx)
    0.00 :          1ceec:       jbe    1ce48 <png_write_row+0x248>
    0.00 :          1cef2:       jmpq   1cc68 <png_write_row+0x68>
    0.00 :          1cef7:       nopw   0x0(%rax,%rax,1)
    0.00 :          1cf00:       testb  $0x7,0x1ec(%rbx)
    0.00 :          1cf07:       je     1cc68 <png_write_row+0x68>
    0.00 :          1cf0d:       nopl   (%rax)
    0.00 :          1cf10:       jmpq   1ce48 <png_write_row+0x248>
    0.00 :          1cf15:       nopl   (%rax)
    0.00 :          1cf18:       testb  $0x1,0x1ec(%rbx)
    0.00 :          1cf1f:       jne    1cc68 <png_write_row+0x68>
    0.00 :          1cf25:       jmpq   1ce48 <png_write_row+0x248>
    0.00 :          1cf2a:       lea    0x8d97(%rip),%rsi        # 25cc8 <png_write_chunk+0x6898>
    0.00 :          1cf31:       mov    %rbx,%rdi
    0.00 :          1cf34:       callq  5500 <png_error@plt>
    0.00 :          1cf39:       lea    0x8d50(%rip),%rsi        # 25c90 <png_write_chunk+0x6860>
    0.00 :          1cf40:       callq  5500 <png_error@plt>
 Percent |       Source code & Disassembly of libpng15.so.15.13.0
-----------------------------------------------
         :
         :
         :
         :       Disassembly of section .text:
         :
         :       000000000001cf50 <png_write_image>:
    0.00 :          1cf50:       push   %r15
    0.00 :          1cf52:       push   %r14
    0.00 :          1cf54:       push   %r13
    0.00 :          1cf56:       push   %r12
    0.00 :          1cf58:       mov    %rdi,%r12
    0.00 :          1cf5b:       push   %rbp
    0.00 :          1cf5c:       push   %rbx
    0.00 :          1cf5d:       sub    $0x8,%rsp
    0.00 :          1cf61:       test   %rdi,%rdi
    0.00 :          1cf64:       je     1cfbe <png_write_image+0x6e>
    0.00 :          1cf66:       mov    %rsi,%r15
    0.00 :          1cf69:       xor    %r13d,%r13d
    0.00 :          1cf6c:       callq  5490 <png_set_interlace_handling@plt>
    0.00 :          1cf71:       test   %eax,%eax
    0.00 :          1cf73:       mov    %eax,%r14d
    0.00 :          1cf76:       jle    1cfbe <png_write_image+0x6e>
    0.00 :          1cf78:       nopl   0x0(%rax,%rax,1)
    0.00 :          1cf80:       mov    0x1d4(%r12),%r10d
    0.00 :          1cf88:       xor    %ebx,%ebx
    0.00 :          1cf8a:       mov    %r15,%rbp
    0.00 :          1cf8d:       test   %r10d,%r10d
    0.00 :          1cf90:       je     1cfb5 <png_write_image+0x65>
    0.00 :          1cf92:       nopw   0x0(%rax,%rax,1)
    0.00 :          1cf98:       mov    0x0(%rbp),%rsi
  100.00 :          1cf9c:       mov    %r12,%rdi
    0.00 :          1cf9f:       add    $0x1,%ebx
    0.00 :          1cfa2:       add    $0x8,%rbp
    0.00 :          1cfa6:       callq  5420 <png_write_row@plt>
    0.00 :          1cfab:       cmp    %ebx,0x1d4(%r12)
    0.00 :          1cfb3:       ja     1cf98 <png_write_image+0x48>
    0.00 :          1cfb5:       add    $0x1,%r13d
    0.00 :          1cfb9:       cmp    %r14d,%r13d
```

```
    0.00 :          1cfbc:       jne    1cf80 <png_write_image+0x30>
    0.00 :          1cfbe:       add    $0x8,%rsp
    0.00 :          1cfc2:       pop    %rbx
    0.00 :          1cfc3:       pop    %rbp
    0.00 :          1cfc4:       pop    %r12
    0.00 :          1cfc6:       pop    %r13
    0.00 :          1cfc8:       pop    %r14
    0.00 :          1cfca:       pop    %r15
    0.00 :          1cfcc:       retq
 Percent |       Source code & Disassembly of libc-2.17.so
-----------------------------------------------
         :
         :
         :
         :       Disassembly of section .text:
         :
         :       000000000007e530 <__libc_free>:
    0.00 :          7e530:       mov    0x328981(%rip),%rax        # 3a6eb8 <_DYNAMIC+0x378>
    0.00 :          7e537:       mov    (%rax),%rax
    0.00 :          7e53a:       test   %rax,%rax
    0.00 :          7e53d:       jne    7e5b2 <__libc_free+0x82>
  100.00 :          7e53f:       test   %rdi,%rdi
    0.00 :          7e542:       je     7e5b0 <__libc_free+0x80>
    0.00 :          7e544:       mov    -0x8(%rdi),%rax
    0.00 :          7e548:       lea    -0x10(%rdi),%rsi
    0.00 :          7e54c:       test   $0x2,%al
    0.00 :          7e54e:       jne    7e570 <__libc_free+0x40>
    0.00 :          7e550:       test   $0x4,%al
    0.00 :          7e552:       lea    0x3290e7(%rip),%rdi        # 3a7640 <main_arena>
    0.00 :          7e559:       je     7e567 <__libc_free+0x37>
    0.00 :          7e55b:       mov    %rsi,%rax
    0.00 :          7e55e:       and    $0xfffffffffc000000,%rax
    0.00 :          7e564:       mov    (%rax),%rdi
    0.00 :          7e567:       xor    %edx,%edx
    0.00 :          7e569:       jmpq   7b190 <_int_free>
    0.00 :          7e56e:       xchg   %ax,%ax
    0.00 :          7e570:       mov    0x328c1e(%rip),%ecx        # 3a7194 <mp_+0x34>
    0.00 :          7e576:       test   %ecx,%ecx
    0.00 :          7e578:       jne    7e5a0 <__libc_free+0x70>
    0.00 :          7e57a:       cmp    0x328bef(%rip),%rax        # 3a7170 <mp_+0x10>
    0.00 :          7e581:       jbe    7e5a0 <__libc_free+0x70>
    0.00 :          7e583:       cmp    $0x2000000,%rax
    0.00 :          7e589:       ja     7e5a0 <__libc_free+0x70>
    0.00 :          7e58b:       and    $0xfffffffffffffff8,%rax
    0.00 :          7e58f:       mov    %rax,0x328bda(%rip)        # 3a7170 <mp_+0x10>
    0.00 :          7e596:       add    %rax,%rax
    0.00 :          7e599:       mov    %rax,0x328bc0(%rip)        # 3a7160 <mp_>
    0.00 :          7e5a0:       mov    %rsi,%rdi
    0.00 :          7e5a3:       jmpq   7b110 <munmap_chunk>
    0.00 :          7e5a8:       nopl   0x0(%rax,%rax,1)
    0.00 :          7e5b0:       repz retq
    0.00 :          7e5b2:       mov    (%rsp),%rsi
    0.00 :          7e5b6:       jmpq   *%rax
 Percent |       Source code & Disassembly of libc-2.17.so
-----------------------------------------------
         :
         :
         :
         :       Disassembly of section .text:
         :
         :       0000000000082660 <__GI_strcmp>:
    0.00 :          82660:       mov    %esi,%ecx
    0.00 :          82662:       mov    %edi,%eax
    0.00 :          82664:       and    $0x3f,%rcx
```

```
 0.00 :        82668:       and    $0x3f,%rax
 0.00 :        8266c:       cmp    $0x30,%ecx
 0.00 :        8266f:       ja     826b0 <__GI_strcmp+0x50>
 0.00 :        82671:       cmp    $0x30,%eax
 0.00 :        82674:       ja     826b0 <__GI_strcmp+0x50>
 0.00 :        82676:       movlpd (%rdi),%xmm1
 0.00 :        8267a:       movlpd (%rsi),%xmm2
 0.00 :        8267e:       movhpd 0x8(%rdi),%xmm1
 0.00 :        82683:       movhpd 0x8(%rsi),%xmm2
 0.00 :        82688:       pxor   %xmm0,%xmm0
 0.00 :        8268c:       pcmpeqb %xmm1,%xmm0
 0.00 :        82690:       pcmpeqb %xmm2,%xmm1
 0.00 :        82694:       psubb  %xmm0,%xmm1
 0.00 :        82698:       pmovmskb %xmm1,%edx
 0.00 :        8269c:       sub    $0xffff,%edx
 0.00 :        826a2:       jne    83a70 <__GI_strcmp+0x1410>
 0.00 :        826a8:       add    $0x10,%rsi
 0.00 :        826ac:       add    $0x10,%rdi
 0.00 :        826b0:       and    $0xfffffffffffffff0,%rsi
 0.00 :        826b4:       and    $0xfffffffffffffff0,%rdi
 0.00 :        826b8:       mov    $0xffff,%edx
 0.00 :        826bd:       xor    %r8d,%r8d
 0.00 :        826c0:       and    $0xf,%ecx
 0.00 :        826c3:       and    $0xf,%eax
 0.00 :        826c6:       cmp    %eax,%ecx
 0.00 :        826c8:       je     826f0 <__GI_strcmp+0x90>
 0.00 :        826ca:       ja     826d3 <__GI_strcmp+0x73>
 0.00 :        826cc:       mov    %edx,%r8d
 0.00 :        826cf:       xchg   %eax,%ecx
 0.00 :        826d0:       xchg   %rsi,%rdi
 0.00 :        826d3:       lea    0xf(%rax),%r9
 0.00 :        826d7:       sub    %rcx,%r9
 0.00 :        826da:       lea    0xe08cf(%rip),%r10        # 162fb0 <CSWTCH.21+0x70>
 0.00 :        826e1:       movslq (%r10,%r9,4),%r9
 0.00 :        826e5:       lea    (%r10,%r9,1),%r10
 0.00 :        826e9:       jmpq   *%r10
 0.00 :        826ec:       nopl   0x0(%rax)
 0.00 :        826f0:       movdqa (%rsi),%xmm1
 0.00 :        826f4:       pxor   %xmm0,%xmm0
 0.00 :        826f8:       pcmpeqb %xmm1,%xmm0
 0.00 :        826fc:       pcmpeqb (%rdi),%xmm1
 0.00 :        82700:       psubb  %xmm0,%xmm1
 0.00 :        82704:       pmovmskb %xmm1,%r9d
 0.00 :        82709:       shr    %cl,%edx
 0.00 :        8270b:       shr    %cl,%r9d
 0.00 :        8270e:       sub    %r9d,%edx
 0.00 :        82711:       jne    83a55 <__GI_strcmp+0x13f5>
 0.00 :        82717:       mov    $0x10,%rcx
 0.00 :        8271e:       mov    $0x10,%r9
 0.00 :        82725:       pxor   %xmm0,%xmm0
 0.00 :        82729:       nopl   0x0(%rax)
 0.00 :        82730:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :        82735:       movdqa (%rdi,%rcx,1),%xmm2
 0.00 :        8273a:       pcmpeqb %xmm1,%xmm0
 0.00 :        8273e:       pcmpeqb %xmm2,%xmm1
 0.00 :        82742:       psubb  %xmm0,%xmm1
 0.00 :        82746:       pmovmskb %xmm1,%edx
 0.00 :        8274a:       sub    $0xffff,%edx
 0.00 :        82750:       jne    83a50 <__GI_strcmp+0x13f0>
 0.00 :        82756:       add    $0x10,%rcx
 0.00 :        8275a:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :        8275f:       movdqa (%rdi,%rcx,1),%xmm2
 0.00 :        82764:       pcmpeqb %xmm1,%xmm0
 0.00 :        82768:       pcmpeqb %xmm2,%xmm1
```

```
 0.00 :        8276c:       psubb  %xmm0,%xmm1
 0.00 :        82770:       pmovmskb %xmm1,%edx
 0.00 :        82774:       sub    $0xffff,%edx
 0.00 :        8277a:       jne    83a50 <__GI_strcmp+0x13f0>
 0.00 :        82780:       add    $0x10,%rcx
 0.00 :        82784:       jmp    82730 <__GI_strcmp+0xd0>
 0.00 :        82786:       nopw   %cs:0x0(%rax,%rax,1)
 0.00 :        82790:       pxor   %xmm0,%xmm0
 0.00 :        82794:       movdqa (%rdi),%xmm2
 0.00 :        82798:       movdqa (%rsi),%xmm1
 0.00 :        8279c:       pcmpeqb %xmm1,%xmm0
 0.00 :        827a0:       pslldq $0xf,%xmm2
 0.00 :        827a5:       pcmpeqb %xmm1,%xmm2
 0.00 :        827a9:       psubb  %xmm0,%xmm2
 0.00 :        827ad:       pmovmskb %xmm2,%r9d
 0.00 :        827b2:       shr    %cl,%edx
 0.00 :        827b4:       shr    %cl,%r9d
 0.00 :        827b7:       sub    %r9d,%edx
 0.00 :        827ba:       jne    83a55 <__GI_strcmp+0x13f5>
 0.00 :        827c0:       movdqa (%rdi),%xmm3
 0.00 :        827c4:       pxor   %xmm0,%xmm0
 0.00 :        827c8:       mov    $0x10,%rcx
 0.00 :        827cf:       mov    $0x1,%r9d
 0.00 :        827d5:       lea    0x1(%rdi),%r10
 0.00 :        827d9:       and    $0xfff,%r10
 0.00 :        827e0:       sub    $0x1000,%r10
 0.00 :        827e7:       nopw   0x0(%rax,%rax,1)
 0.00 :        827f0:       add    $0x10,%r10
 0.00 :        827f4:       jg     82890 <__GI_strcmp+0x230>
 0.00 :        827fa:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :        827ff:       movdqa (%rdi,%rcx,1),%xmm2
 0.00 :        82804:       movdqa %xmm2,%xmm4
 0.00 :        82808:       psrldq $0x1,%xmm3
 0.00 :        8280d:       pslldq $0xf,%xmm2
 0.00 :        82812:       por    %xmm3,%xmm2
 0.00 :        82816:       pcmpeqb %xmm1,%xmm0
 0.00 :        8281a:       pcmpeqb %xmm2,%xmm1
 0.00 :        8281e:       psubb  %xmm0,%xmm1
 0.00 :        82822:       pmovmskb %xmm1,%edx
 0.00 :        82826:       sub    $0xffff,%edx
 0.00 :        8282c:       jne    83a50 <__GI_strcmp+0x13f0>
 0.00 :        82832:       add    $0x10,%rcx
 0.00 :        82836:       movdqa %xmm4,%xmm3
 0.00 :        8283a:       add    $0x10,%r10
 0.00 :        8283e:       jg     82890 <__GI_strcmp+0x230>
 0.00 :        82840:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :        82845:       movdqa (%rdi,%rcx,1),%xmm2
 0.00 :        8284a:       movdqa %xmm2,%xmm4
 0.00 :        8284e:       psrldq $0x1,%xmm3
 0.00 :        82853:       pslldq $0xf,%xmm2
 0.00 :        82858:       por    %xmm3,%xmm2
 0.00 :        8285c:       pcmpeqb %xmm1,%xmm0
 0.00 :        82860:       pcmpeqb %xmm2,%xmm1
 0.00 :        82864:       psubb  %xmm0,%xmm1
 0.00 :        82868:       pmovmskb %xmm1,%edx
 0.00 :        8286c:       sub    $0xffff,%edx
 0.00 :        82872:       jne    83a50 <__GI_strcmp+0x13f0>
 0.00 :        82878:       add    $0x10,%rcx
 0.00 :        8287c:       movdqa %xmm4,%xmm3
 0.00 :        82880:       jmpq   827f0 <__GI_strcmp+0x190>
 0.00 :        82885:       data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :        82890:       pcmpeqb %xmm3,%xmm0
 0.00 :        82894:       pmovmskb %xmm0,%edx
 0.00 :        82898:       test   $0xfffe,%edx
```

```
 0.00 :        8289e:       jne    828b0 <__GI_strcmp+0x250>
 0.00 :        828a0:       pxor   %xmm0,%xmm0
 0.00 :        828a4:       sub    $0x1000,%r10
 0.00 :        828ab:       jmpq   827fa <__GI_strcmp+0x19a>
 0.00 :        828b0:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :        828b5:       psrldq $0x1,%xmm0
 0.00 :        828ba:       psrldq $0x1,%xmm3
 0.00 :        828bf:       jmpq   83a40 <__GI_strcmp+0x13e0>
 0.00 :        828c4:       data32 data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :        828d0:       pxor   %xmm0,%xmm0
 0.00 :        828d4:       movdqa (%rdi),%xmm2
 0.00 :        828d8:       movdqa (%rsi),%xmm1
 0.00 :        828dc:       pcmpeqb %xmm1,%xmm0
 0.00 :        828e0:       pslldq $0xe,%xmm2
 0.00 :        828e5:       pcmpeqb %xmm1,%xmm2
 0.00 :        828e9:       psubb  %xmm0,%xmm2
 0.00 :        828ed:       pmovmskb %xmm2,%r9d
 0.00 :        828f2:       shr    %cl,%edx
 0.00 :        828f4:       shr    %cl,%r9d
 0.00 :        828f7:       sub    %r9d,%edx
 0.00 :        828fa:       jne    83a55 <__GI_strcmp+0x13f5>
 0.00 :        82900:       movdqa (%rdi),%xmm3
 0.00 :        82904:       pxor   %xmm0,%xmm0
 0.00 :        82908:       mov    $0x10,%rcx
 0.00 :        8290f:       mov    $0x2,%r9d
 0.00 :        82915:       lea    0x2(%rdi),%r10
 0.00 :        82919:       and    $0xfff,%r10
 0.00 :        82920:       sub    $0x1000,%r10
 0.00 :        82927:       nopw   0x0(%rax,%rax,1)
 0.00 :        82930:       add    $0x10,%r10
 0.00 :        82934:       jg     829d0 <__GI_strcmp+0x370>
 0.00 :        8293a:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :        8293f:       movdqa (%rdi,%rcx,1),%xmm2
 0.00 :        82944:       movdqa %xmm2,%xmm4
 0.00 :        82948:       psrldq $0x2,%xmm3
 0.00 :        8294d:       pslldq $0xe,%xmm2
 0.00 :        82952:       por    %xmm3,%xmm2
 0.00 :        82956:       pcmpeqb %xmm1,%xmm0
 0.00 :        8295a:       pcmpeqb %xmm2,%xmm1
 0.00 :        8295e:       psubb  %xmm0,%xmm1
 0.00 :        82962:       pmovmskb %xmm1,%edx
 0.00 :        82966:       sub    $0xffff,%edx
 0.00 :        8296c:       jne    83a50 <__GI_strcmp+0x13f0>
 0.00 :        82972:       add    $0x10,%rcx
 0.00 :        82976:       movdqa %xmm4,%xmm3
 0.00 :        8297a:       add    $0x10,%r10
 0.00 :        8297e:       jg     829d0 <__GI_strcmp+0x370>
 0.00 :        82980:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :        82985:       movdqa (%rdi,%rcx,1),%xmm2
 0.00 :        8298a:       movdqa %xmm2,%xmm4
 0.00 :        8298e:       psrldq $0x2,%xmm3
 0.00 :        82993:       pslldq $0xe,%xmm2
 0.00 :        82998:       por    %xmm3,%xmm2
 0.00 :        8299c:       pcmpeqb %xmm1,%xmm0
 0.00 :        829a0:       pcmpeqb %xmm2,%xmm1
 0.00 :        829a4:       psubb  %xmm0,%xmm1
 0.00 :        829a8:       pmovmskb %xmm1,%edx
 0.00 :        829ac:       sub    $0xffff,%edx
 0.00 :        829b2:       jne    83a50 <__GI_strcmp+0x13f0>
 0.00 :        829b8:       add    $0x10,%rcx
 0.00 :        829bc:       movdqa %xmm4,%xmm3
 0.00 :        829c0:       jmpq   82930 <__GI_strcmp+0x2d0>
 0.00 :        829c5:       data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :        829d0:       pcmpeqb %xmm3,%xmm0
```

```
 0.00 :        829d4:       pmovmskb %xmm0,%edx
 0.00 :        829d8:       test   $0xfffc,%edx
 0.00 :        829de:       jne    829f0 <__GI_strcmp+0x390>
 0.00 :        829e0:       pxor   %xmm0,%xmm0
 0.00 :        829e4:       sub    $0x1000,%r10
 0.00 :        829eb:       jmpq   8293a <__GI_strcmp+0x2da>
 0.00 :        829f0:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :        829f5:       psrldq $0x2,%xmm0
 0.00 :        829fa:       psrldq $0x2,%xmm3
 0.00 :        829ff:       jmpq   83a40 <__GI_strcmp+0x13e0>
 0.00 :        82a04:       data32 data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :        82a10:       pxor   %xmm0,%xmm0
 0.00 :        82a14:       movdqa (%rdi),%xmm2
 0.00 :        82a18:       movdqa (%rsi),%xmm1
 0.00 :        82a1c:       pcmpeqb %xmm1,%xmm0
 0.00 :        82a20:       pslldq $0xd,%xmm2
 0.00 :        82a25:       pcmpeqb %xmm1,%xmm2
 0.00 :        82a29:       psubb  %xmm0,%xmm2
 0.00 :        82a2d:       pmovmskb %xmm2,%r9d
 0.00 :        82a32:       shr    %cl,%edx
 0.00 :        82a34:       shr    %cl,%r9d
 0.00 :        82a37:       sub    %r9d,%edx
 0.00 :        82a3a:       jne    83a55 <__GI_strcmp+0x13f5>
 0.00 :        82a40:       movdqa (%rdi),%xmm3
 0.00 :        82a44:       pxor   %xmm0,%xmm0
 0.00 :        82a48:       mov    $0x10,%rcx
 0.00 :        82a4f:       mov    $0x3,%r9d
 0.00 :        82a55:       lea    0x3(%rdi),%r10
 0.00 :        82a59:       and    $0xfff,%r10
 0.00 :        82a60:       sub    $0x1000,%r10
 0.00 :        82a67:       nopw   0x0(%rax,%rax,1)
 0.00 :        82a70:       add    $0x10,%r10
 0.00 :        82a74:       jg     82b10 <__GI_strcmp+0x4b0>
 0.00 :        82a7a:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :        82a7f:       movdqa (%rdi,%rcx,1),%xmm2
 0.00 :        82a84:       movdqa %xmm2,%xmm4
 0.00 :        82a88:       psrldq $0x3,%xmm3
 0.00 :        82a8d:       pslldq $0xd,%xmm2
 0.00 :        82a92:       por    %xmm3,%xmm2
 0.00 :        82a96:       pcmpeqb %xmm1,%xmm0
 0.00 :        82a9a:       pcmpeqb %xmm2,%xmm1
 0.00 :        82a9e:       psubb  %xmm0,%xmm1
 0.00 :        82aa2:       pmovmskb %xmm1,%edx
 0.00 :        82aa6:       sub    $0xffff,%edx
 0.00 :        82aac:       jne    83a50 <__GI_strcmp+0x13f0>
 0.00 :        82ab2:       add    $0x10,%rcx
 0.00 :        82ab6:       movdqa %xmm4,%xmm3
 0.00 :        82aba:       add    $0x10,%r10
 0.00 :        82abe:       jg     82b10 <__GI_strcmp+0x4b0>
 0.00 :        82ac0:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :        82ac5:       movdqa (%rdi,%rcx,1),%xmm2
 0.00 :        82aca:       movdqa %xmm2,%xmm4
 0.00 :        82ace:       psrldq $0x3,%xmm3
 0.00 :        82ad3:       pslldq $0xd,%xmm2
 0.00 :        82ad8:       por    %xmm3,%xmm2
 0.00 :        82adc:       pcmpeqb %xmm1,%xmm0
 0.00 :        82ae0:       pcmpeqb %xmm2,%xmm1
 0.00 :        82ae4:       psubb  %xmm0,%xmm1
 0.00 :        82ae8:       pmovmskb %xmm1,%edx
 0.00 :        82aec:       sub    $0xffff,%edx
 0.00 :        82af2:       jne    83a50 <__GI_strcmp+0x13f0>
 0.00 :        82af8:       add    $0x10,%rcx
 0.00 :        82afc:       movdqa %xmm4,%xmm3
 0.00 :        82b00:       jmpq   82a70 <__GI_strcmp+0x410>
```

```
 0.00 :        82b05:       data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :        82b10:       pcmpeqb %xmm3,%xmm0
 0.00 :        82b14:       pmovmskb %xmm0,%edx
 0.00 :        82b18:       test   $0xfff8,%edx
 0.00 :        82b1e:       jne    82b30 <__GI_strcmp+0x4d0>
 0.00 :        82b20:       pxor   %xmm0,%xmm0
 0.00 :        82b24:       sub    $0x1000,%r10
 0.00 :        82b2b:       jmpq   82a7a <__GI_strcmp+0x41a>
 0.00 :        82b30:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :        82b35:       psrldq $0x3,%xmm0
 0.00 :        82b3a:       psrldq $0x3,%xmm3
 0.00 :        82b3f:       jmpq   83a40 <__GI_strcmp+0x13e0>
 0.00 :        82b44:       data32 data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :        82b50:       pxor   %xmm0,%xmm0
 0.00 :        82b54:       movdqa (%rdi),%xmm2
 0.00 :        82b58:       movdqa (%rsi),%xmm1
 0.00 :        82b5c:       pcmpeqb %xmm1,%xmm0
 0.00 :        82b60:       pslldq $0xc,%xmm2
 0.00 :        82b65:       pcmpeqb %xmm1,%xmm2
 0.00 :        82b69:       psubb  %xmm0,%xmm2
 0.00 :        82b6d:       pmovmskb %xmm2,%r9d
 0.00 :        82b72:       shr    %cl,%edx
 0.00 :        82b74:       shr    %cl,%r9d
 0.00 :        82b77:       sub    %r9d,%edx
 0.00 :        82b7a:       jne    83a55 <__GI_strcmp+0x13f5>
 0.00 :        82b80:       movdqa (%rdi),%xmm3
 0.00 :        82b84:       pxor   %xmm0,%xmm0
 0.00 :        82b88:       mov    $0x10,%rcx
 0.00 :        82b8f:       mov    $0x4,%r9d
 0.00 :        82b95:       lea    0x4(%rdi),%r10
 0.00 :        82b99:       and    $0xfff,%r10
 0.00 :        82ba0:       sub    $0x1000,%r10
 0.00 :        82ba7:       nopw   0x0(%rax,%rax,1)
 0.00 :        82bb0:       add    $0x10,%r10
 0.00 :        82bb4:       jg     82c50 <__GI_strcmp+0x5f0>
 0.00 :        82bba:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :        82bbf:       movdqa (%rdi,%rcx,1),%xmm2
 0.00 :        82bc4:       movdqa %xmm2,%xmm4
 0.00 :        82bc8:       psrldq $0x4,%xmm3
 0.00 :        82bcd:       pslldq $0xc,%xmm2
 0.00 :        82bd2:       por    %xmm3,%xmm2
 0.00 :        82bd6:       pcmpeqb %xmm1,%xmm0
 0.00 :        82bda:       pcmpeqb %xmm2,%xmm1
 0.00 :        82bde:       psubb  %xmm0,%xmm1
 0.00 :        82be2:       pmovmskb %xmm1,%edx
 0.00 :        82be6:       sub    $0xffff,%edx
 0.00 :        82bec:       jne    83a50 <__GI_strcmp+0x13f0>
 0.00 :        82bf2:       add    $0x10,%rcx
 0.00 :        82bf6:       movdqa %xmm4,%xmm3
 0.00 :        82bfa:       add    $0x10,%r10
 0.00 :        82bfe:       jg     82c50 <__GI_strcmp+0x5f0>
 0.00 :        82c00:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :        82c05:       movdqa (%rdi,%rcx,1),%xmm2
 0.00 :        82c0a:       movdqa %xmm2,%xmm4
 0.00 :        82c0e:       psrldq $0x4,%xmm3
 0.00 :        82c13:       pslldq $0xc,%xmm2
 0.00 :        82c18:       por    %xmm3,%xmm2
 0.00 :        82c1c:       pcmpeqb %xmm1,%xmm0
 0.00 :        82c20:       pcmpeqb %xmm2,%xmm1
 0.00 :        82c24:       psubb  %xmm0,%xmm1
 0.00 :        82c28:       pmovmskb %xmm1,%edx
 0.00 :        82c2c:       sub    $0xffff,%edx
 0.00 :        82c32:       jne    83a50 <__GI_strcmp+0x13f0>
 0.00 :        82c38:       add    $0x10,%rcx
```

```
 0.00 :        82c3c:       movdqa %xmm4,%xmm3
 0.00 :        82c40:       jmpq   82bb0 <__GI_strcmp+0x550>
 0.00 :        82c45:       data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :        82c50:       pcmpeqb %xmm3,%xmm0
 0.00 :        82c54:       pmovmskb %xmm0,%edx
 0.00 :        82c58:       test   $0xfff0,%edx
 0.00 :        82c5e:       jne    82c70 <__GI_strcmp+0x610>
 0.00 :        82c60:       pxor   %xmm0,%xmm0
 0.00 :        82c64:       sub    $0x1000,%r10
 0.00 :        82c6b:       jmpq   82bba <__GI_strcmp+0x55a>
 0.00 :        82c70:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :        82c75:       psrldq $0x4,%xmm0
 0.00 :        82c7a:       psrldq $0x4,%xmm3
 0.00 :        82c7f:       jmpq   83a40 <__GI_strcmp+0x13e0>
 0.00 :        82c84:       data32 data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :        82c90:       pxor   %xmm0,%xmm0
 0.00 :        82c94:       movdqa (%rdi),%xmm2
 0.00 :        82c98:       movdqa (%rsi),%xmm1
 0.00 :        82c9c:       pcmpeqb %xmm1,%xmm0
 0.00 :        82ca0:       pslldq $0xb,%xmm2
 0.00 :        82ca5:       pcmpeqb %xmm1,%xmm2
 0.00 :        82ca9:       psubb  %xmm0,%xmm2
 0.00 :        82cad:       pmovmskb %xmm2,%r9d
 0.00 :        82cb2:       shr    %cl,%edx
 0.00 :        82cb4:       shr    %cl,%r9d
 0.00 :        82cb7:       sub    %r9d,%edx
 0.00 :        82cba:       jne    83a55 <__GI_strcmp+0x13f5>
 0.00 :        82cc0:       movdqa (%rdi),%xmm3
 0.00 :        82cc4:       pxor   %xmm0,%xmm0
 0.00 :        82cc8:       mov    $0x10,%rcx
 0.00 :        82ccf:       mov    $0x5,%r9d
 0.00 :        82cd5:       lea    0x5(%rdi),%r10
 0.00 :        82cd9:       and    $0xfff,%r10
 0.00 :        82ce0:       sub    $0x1000,%r10
 0.00 :        82ce7:       nopw   0x0(%rax,%rax,1)
 0.00 :        82cf0:       add    $0x10,%r10
 0.00 :        82cf4:       jg     82d90 <__GI_strcmp+0x730>
 0.00 :        82cfa:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :        82cff:       movdqa (%rdi,%rcx,1),%xmm2
 0.00 :        82d04:       movdqa %xmm2,%xmm4
 0.00 :        82d08:       psrldq $0x5,%xmm3
 0.00 :        82d0d:       pslldq $0xb,%xmm2
 0.00 :        82d12:       por    %xmm3,%xmm2
 0.00 :        82d16:       pcmpeqb %xmm1,%xmm0
 0.00 :        82d1a:       pcmpeqb %xmm2,%xmm1
 0.00 :        82d1e:       psubb  %xmm0,%xmm1
 0.00 :        82d22:       pmovmskb %xmm1,%edx
 0.00 :        82d26:       sub    $0xffff,%edx
 0.00 :        82d2c:       jne    83a50 <__GI_strcmp+0x13f0>
 0.00 :        82d32:       add    $0x10,%rcx
 0.00 :        82d36:       movdqa %xmm4,%xmm3
 0.00 :        82d3a:       add    $0x10,%r10
 0.00 :        82d3e:       jg     82d90 <__GI_strcmp+0x730>
 0.00 :        82d40:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :        82d45:       movdqa (%rdi,%rcx,1),%xmm2
 0.00 :        82d4a:       movdqa %xmm2,%xmm4
 0.00 :        82d4e:       psrldq $0x5,%xmm3
 0.00 :        82d53:       pslldq $0xb,%xmm2
 0.00 :        82d58:       por    %xmm3,%xmm2
 0.00 :        82d5c:       pcmpeqb %xmm1,%xmm0
 0.00 :        82d60:       pcmpeqb %xmm2,%xmm1
 0.00 :        82d64:       psubb  %xmm0,%xmm1
 0.00 :        82d68:       pmovmskb %xmm1,%edx
 0.00 :        82d6c:       sub    $0xffff,%edx
```

```
 0.00 :        82d72:       jne    83a50 <__GI_strcmp+0x13f0>
 0.00 :        82d78:       add    $0x10,%rcx
 0.00 :        82d7c:       movdqa %xmm4,%xmm3
 0.00 :        82d80:       jmpq   82cf0 <__GI_strcmp+0x690>
 0.00 :        82d85:       data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :        82d90:       pcmpeqb %xmm3,%xmm0
 0.00 :        82d94:       pmovmskb %xmm0,%edx
 0.00 :        82d98:       test   $0xffe0,%edx
 0.00 :        82d9e:       jne    82db0 <__GI_strcmp+0x750>
 0.00 :        82da0:       pxor   %xmm0,%xmm0
 0.00 :        82da4:       sub    $0x1000,%r10
 0.00 :        82dab:       jmpq   82cfa <__GI_strcmp+0x69a>
 0.00 :        82db0:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :        82db5:       psrldq $0x5,%xmm0
 0.00 :        82dba:       psrldq $0x5,%xmm3
 0.00 :        82dbf:       jmpq   83a40 <__GI_strcmp+0x13e0>
 0.00 :        82dc4:       data32 data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :        82dd0:       pxor   %xmm0,%xmm0
 0.00 :        82dd4:       movdqa (%rdi),%xmm2
 0.00 :        82dd8:       movdqa (%rsi),%xmm1
 0.00 :        82ddc:       pcmpeqb %xmm1,%xmm0
 0.00 :        82de0:       pslldq $0xa,%xmm2
 0.00 :        82de5:       pcmpeqb %xmm1,%xmm2
 0.00 :        82de9:       psubb  %xmm0,%xmm2
 0.00 :        82ded:       pmovmskb %xmm2,%r9d
 0.00 :        82df2:       shr    %cl,%edx
 0.00 :        82df4:       shr    %cl,%r9d
 0.00 :        82df7:       sub    %r9d,%edx
 0.00 :        82dfa:       jne    83a55 <__GI_strcmp+0x13f5>
 0.00 :        82e00:       movdqa (%rdi),%xmm3
 0.00 :        82e04:       pxor   %xmm0,%xmm0
 0.00 :        82e08:       mov    $0x10,%rcx
 0.00 :        82e0f:       mov    $0x6,%r9d
 0.00 :        82e15:       lea    0x6(%rdi),%r10
 0.00 :        82e19:       and    $0xfff,%r10
 0.00 :        82e20:       sub    $0x1000,%r10
 0.00 :        82e27:       nopw   0x0(%rax,%rax,1)
 0.00 :        82e30:       add    $0x10,%r10
 0.00 :        82e34:       jg     82ed0 <__GI_strcmp+0x870>
 0.00 :        82e3a:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :        82e3f:       movdqa (%rdi,%rcx,1),%xmm2
 0.00 :        82e44:       movdqa %xmm2,%xmm4
 0.00 :        82e48:       psrldq $0x6,%xmm3
 0.00 :        82e4d:       pslldq $0xa,%xmm2
 0.00 :        82e52:       por    %xmm3,%xmm2
 0.00 :        82e56:       pcmpeqb %xmm1,%xmm0
 0.00 :        82e5a:       pcmpeqb %xmm2,%xmm1
 0.00 :        82e5e:       psubb  %xmm0,%xmm1
 0.00 :        82e62:       pmovmskb %xmm1,%edx
 0.00 :        82e66:       sub    $0xffff,%edx
 0.00 :        82e6c:       jne    83a50 <__GI_strcmp+0x13f0>
 0.00 :        82e72:       add    $0x10,%rcx
 0.00 :        82e76:       movdqa %xmm4,%xmm3
 0.00 :        82e7a:       add    $0x10,%r10
 0.00 :        82e7e:       jg     82ed0 <__GI_strcmp+0x870>
 0.00 :        82e80:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :        82e85:       movdqa (%rdi,%rcx,1),%xmm2
 0.00 :        82e8a:       movdqa %xmm2,%xmm4
 0.00 :        82e8e:       psrldq $0x6,%xmm3
 0.00 :        82e93:       pslldq $0xa,%xmm2
 0.00 :        82e98:       por    %xmm3,%xmm2
 0.00 :        82e9c:       pcmpeqb %xmm1,%xmm0
 0.00 :        82ea0:       pcmpeqb %xmm2,%xmm1
 0.00 :        82ea4:       psubb  %xmm0,%xmm1
```

```
 0.00 :        82ea8:       pmovmskb %xmm1,%edx
 0.00 :        82eac:       sub    $0xffff,%edx
 0.00 :        82eb2:       jne    83a50 <__GI_strcmp+0x13f0>
 0.00 :        82eb8:       add    $0x10,%rcx
 0.00 :        82ebc:       movdqa %xmm4,%xmm3
 0.00 :        82ec0:       jmpq   82e30 <__GI_strcmp+0x7d0>
 0.00 :        82ec5:       data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :        82ed0:       pcmpeqb %xmm3,%xmm0
 0.00 :        82ed4:       pmovmskb %xmm0,%edx
 0.00 :        82ed8:       test   $0xffc0,%edx
 0.00 :        82ede:       jne    82ef0 <__GI_strcmp+0x890>
 0.00 :        82ee0:       pxor   %xmm0,%xmm0
 0.00 :        82ee4:       sub    $0x1000,%r10
 0.00 :        82eeb:       jmpq   82e3a <__GI_strcmp+0x7da>
 0.00 :        82ef0:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :        82ef5:       psrldq $0x6,%xmm0
 0.00 :        82efa:       psrldq $0x6,%xmm3
 0.00 :        82eff:       jmpq   83a40 <__GI_strcmp+0x13e0>
 0.00 :        82f04:       data32 data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :        82f10:       pxor   %xmm0,%xmm0
 0.00 :        82f14:       movdqa (%rdi),%xmm2
 0.00 :        82f18:       movdqa (%rsi),%xmm1
 0.00 :        82f1c:       pcmpeqb %xmm1,%xmm0
 0.00 :        82f20:       pslldq $0x9,%xmm2
 0.00 :        82f25:       pcmpeqb %xmm1,%xmm2
 0.00 :        82f29:       psubb  %xmm0,%xmm2
 0.00 :        82f2d:       pmovmskb %xmm2,%r9d
 0.00 :        82f32:       shr    %cl,%edx
 0.00 :        82f34:       shr    %cl,%r9d
 0.00 :        82f37:       sub    %r9d,%edx
 0.00 :        82f3a:       jne    83a55 <__GI_strcmp+0x13f5>
 0.00 :        82f40:       movdqa (%rdi),%xmm3
 0.00 :        82f44:       pxor   %xmm0,%xmm0
 0.00 :        82f48:       mov    $0x10,%rcx
 0.00 :        82f4f:       mov    $0x7,%r9d
 0.00 :        82f55:       lea    0x7(%rdi),%r10
 0.00 :        82f59:       and    $0xfff,%r10
 0.00 :        82f60:       sub    $0x1000,%r10
 0.00 :        82f67:       nopw   0x0(%rax,%rax,1)
 0.00 :        82f70:       add    $0x10,%r10
 0.00 :        82f74:       jg     83010 <__GI_strcmp+0x9b0>
 0.00 :        82f7a:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :        82f7f:       movdqa (%rdi,%rcx,1),%xmm2
 0.00 :        82f84:       movdqa %xmm2,%xmm4
 0.00 :        82f88:       psrldq $0x7,%xmm3
 0.00 :        82f8d:       pslldq $0x9,%xmm2
 0.00 :        82f92:       por    %xmm3,%xmm2
 0.00 :        82f96:       pcmpeqb %xmm1,%xmm0
 0.00 :        82f9a:       pcmpeqb %xmm2,%xmm1
 0.00 :        82f9e:       psubb  %xmm0,%xmm1
 0.00 :        82fa2:       pmovmskb %xmm1,%edx
 0.00 :        82fa6:       sub    $0xffff,%edx
 0.00 :        82fac:       jne    83a50 <__GI_strcmp+0x13f0>
 0.00 :        82fb2:       add    $0x10,%rcx
 0.00 :        82fb6:       movdqa %xmm4,%xmm3
 0.00 :        82fba:       add    $0x10,%r10
 0.00 :        82fbe:       jg     83010 <__GI_strcmp+0x9b0>
 0.00 :        82fc0:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :        82fc5:       movdqa (%rdi,%rcx,1),%xmm2
 0.00 :        82fca:       movdqa %xmm2,%xmm4
 0.00 :        82fce:       psrldq $0x7,%xmm3
 0.00 :        82fd3:       pslldq $0x9,%xmm2
 0.00 :        82fd8:       por    %xmm3,%xmm2
 0.00 :        82fdc:       pcmpeqb %xmm1,%xmm0
```

```
 0.00 :        82fe0:       pcmpeqb %xmm2,%xmm1
 0.00 :        82fe4:       psubb  %xmm0,%xmm1
 0.00 :        82fe8:       pmovmskb %xmm1,%edx
 0.00 :        82fec:       sub    $0xffff,%edx
 0.00 :        82ff2:       jne    83a50 <__GI_strcmp+0x13f0>
 0.00 :        82ff8:       add    $0x10,%rcx
 0.00 :        82ffc:       movdqa %xmm4,%xmm3
 0.00 :        83000:       jmpq   82f70 <__GI_strcmp+0x910>
 0.00 :        83005:       data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :        83010:       pcmpeqb %xmm3,%xmm0
 0.00 :        83014:       pmovmskb %xmm0,%edx
 0.00 :        83018:       test   $0xff80,%edx
 0.00 :        8301e:       jne    83030 <__GI_strcmp+0x9d0>
 0.00 :        83020:       pxor   %xmm0,%xmm0
 0.00 :        83024:       sub    $0x1000,%r10
 0.00 :        8302b:       jmpq   82f7a <__GI_strcmp+0x91a>
 0.00 :        83030:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :        83035:       psrldq $0x7,%xmm0
 0.00 :        8303a:       psrldq $0x7,%xmm3
 0.00 :        8303f:       jmpq   83a40 <__GI_strcmp+0x13e0>
 0.00 :        83044:       data32 data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :        83050:       pxor   %xmm0,%xmm0
 0.00 :        83054:       movdqa (%rdi),%xmm2
 0.00 :        83058:       movdqa (%rsi),%xmm1
 0.00 :        8305c:       pcmpeqb %xmm1,%xmm0
 0.00 :        83060:       pslldq $0x8,%xmm2
 0.00 :        83065:       pcmpeqb %xmm1,%xmm2
 0.00 :        83069:       psubb  %xmm0,%xmm2
 0.00 :        8306d:       pmovmskb %xmm2,%r9d
 0.00 :        83072:       shr    %cl,%edx
 0.00 :        83074:       shr    %cl,%r9d
 0.00 :        83077:       sub    %r9d,%edx
 0.00 :        8307a:       jne    83a55 <__GI_strcmp+0x13f5>
 0.00 :        83080:       movdqa (%rdi),%xmm3
 0.00 :        83084:       pxor   %xmm0,%xmm0
 0.00 :        83088:       mov    $0x10,%rcx
 0.00 :        8308f:       mov    $0x8,%r9d
 0.00 :        83095:       lea    0x8(%rdi),%r10
 0.00 :        83099:       and    $0xfff,%r10
 0.00 :        830a0:       sub    $0x1000,%r10
 0.00 :        830a7:       nopw   0x0(%rax,%rax,1)
 0.00 :        830b0:       add    $0x10,%r10
 0.00 :        830b4:       jg     83150 <__GI_strcmp+0xaf0>
 0.00 :        830ba:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :        830bf:       movdqa (%rdi,%rcx,1),%xmm2
 0.00 :        830c4:       movdqa %xmm2,%xmm4
 0.00 :        830c8:       psrldq $0x8,%xmm3
 0.00 :        830cd:       pslldq $0x8,%xmm2
 0.00 :        830d2:       por    %xmm3,%xmm2
 0.00 :        830d6:       pcmpeqb %xmm1,%xmm0
 0.00 :        830da:       pcmpeqb %xmm2,%xmm1
 0.00 :        830de:       psubb  %xmm0,%xmm1
 0.00 :        830e2:       pmovmskb %xmm1,%edx
 0.00 :        830e6:       sub    $0xffff,%edx
 0.00 :        830ec:       jne    83a50 <__GI_strcmp+0x13f0>
 0.00 :        830f2:       add    $0x10,%rcx
 0.00 :        830f6:       movdqa %xmm4,%xmm3
 0.00 :        830fa:       add    $0x10,%r10
 0.00 :        830fe:       jg     83150 <__GI_strcmp+0xaf0>
 0.00 :        83100:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :        83105:       movdqa (%rdi,%rcx,1),%xmm2
 0.00 :        8310a:       movdqa %xmm2,%xmm4
 0.00 :        8310e:       psrldq $0x8,%xmm3
 0.00 :        83113:       pslldq $0x8,%xmm2
```

```
 0.00 :        83118:       por    %xmm3,%xmm2
 0.00 :        8311c:       pcmpeqb %xmm1,%xmm0
 0.00 :        83120:       pcmpeqb %xmm2,%xmm1
 0.00 :        83124:       psubb  %xmm0,%xmm1
 0.00 :        83128:       pmovmskb %xmm1,%edx
 0.00 :        8312c:       sub    $0xffff,%edx
 0.00 :        83132:       jne    83a50 <__GI_strcmp+0x13f0>
 0.00 :        83138:       add    $0x10,%rcx
 0.00 :        8313c:       movdqa %xmm4,%xmm3
 0.00 :        83140:       jmpq   830b0 <__GI_strcmp+0xa50>
 0.00 :        83145:       data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :        83150:       pcmpeqb %xmm3,%xmm0
 0.00 :        83154:       pmovmskb %xmm0,%edx
 0.00 :        83158:       test   $0xff00,%edx
 0.00 :        8315e:       jne    83170 <__GI_strcmp+0xb10>
 0.00 :        83160:       pxor   %xmm0,%xmm0
 0.00 :        83164:       sub    $0x1000,%r10
 0.00 :        8316b:       jmpq   830ba <__GI_strcmp+0xa5a>
 0.00 :        83170:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :        83175:       psrldq $0x8,%xmm0
 0.00 :        8317a:       psrldq $0x8,%xmm3
 0.00 :        8317f:       jmpq   83a40 <__GI_strcmp+0x13e0>
 0.00 :        83184:       data32 data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :        83190:       pxor   %xmm0,%xmm0
 0.00 :        83194:       movdqa (%rdi),%xmm2
 0.00 :        83198:       movdqa (%rsi),%xmm1
 0.00 :        8319c:       pcmpeqb %xmm1,%xmm0
 0.00 :        831a0:       pslldq $0x7,%xmm2
 0.00 :        831a5:       pcmpeqb %xmm1,%xmm2
 0.00 :        831a9:       psubb  %xmm0,%xmm2
 0.00 :        831ad:       pmovmskb %xmm2,%r9d
 0.00 :        831b2:       shr    %cl,%edx
 0.00 :        831b4:       shr    %cl,%r9d
 0.00 :        831b7:       sub    %r9d,%edx
 0.00 :        831ba:       jne    83a55 <__GI_strcmp+0x13f5>
 0.00 :        831c0:       movdqa (%rdi),%xmm3
 0.00 :        831c4:       pxor   %xmm0,%xmm0
 0.00 :        831c8:       mov    $0x10,%rcx
 0.00 :        831cf:       mov    $0x9,%r9d
 0.00 :        831d5:       lea    0x9(%rdi),%r10
 0.00 :        831d9:       and    $0xfff,%r10
 0.00 :        831e0:       sub    $0x1000,%r10
 0.00 :        831e7:       nopw   0x0(%rax,%rax,1)
 0.00 :        831f0:       add    $0x10,%r10
 0.00 :        831f4:       jg     83290 <__GI_strcmp+0xc30>
 0.00 :        831fa:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :        831ff:       movdqa (%rdi,%rcx,1),%xmm2
 0.00 :        83204:       movdqa %xmm2,%xmm4
 0.00 :        83208:       psrldq $0x9,%xmm3
 0.00 :        8320d:       pslldq $0x7,%xmm2
 0.00 :        83212:       por    %xmm3,%xmm2
 0.00 :        83216:       pcmpeqb %xmm1,%xmm0
 0.00 :        8321a:       pcmpeqb %xmm2,%xmm1
 0.00 :        8321e:       psubb  %xmm0,%xmm1
 0.00 :        83222:       pmovmskb %xmm1,%edx
 0.00 :        83226:       sub    $0xffff,%edx
 0.00 :        8322c:       jne    83a50 <__GI_strcmp+0x13f0>
 0.00 :        83232:       add    $0x10,%rcx
 0.00 :        83236:       movdqa %xmm4,%xmm3
 0.00 :        8323a:       add    $0x10,%r10
 0.00 :        8323e:       jg     83290 <__GI_strcmp+0xc30>
 0.00 :        83240:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :        83245:       movdqa (%rdi,%rcx,1),%xmm2
 0.00 :        8324a:       movdqa %xmm2,%xmm4
```

```
 0.00 :        8324e:       psrldq $0x9,%xmm3
 0.00 :        83253:       pslldq $0x7,%xmm2
 0.00 :        83258:       por    %xmm3,%xmm2
 0.00 :        8325c:       pcmpeqb %xmm1,%xmm0
 0.00 :        83260:       pcmpeqb %xmm2,%xmm1
 0.00 :        83264:       psubb  %xmm0,%xmm1
 0.00 :        83268:       pmovmskb %xmm1,%edx
 0.00 :        8326c:       sub    $0xffff,%edx
 0.00 :        83272:       jne    83a50 <__GI_strcmp+0x13f0>
 0.00 :        83278:       add    $0x10,%rcx
 0.00 :        8327c:       movdqa %xmm4,%xmm3
 0.00 :        83280:       jmpq   831f0 <__GI_strcmp+0xb90>
 0.00 :        83285:       data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :        83290:       pcmpeqb %xmm3,%xmm0
 0.00 :        83294:       pmovmskb %xmm0,%edx
 0.00 :        83298:       test   $0xfe00,%edx
 0.00 :        8329e:       jne    832b0 <__GI_strcmp+0xc50>
 0.00 :        832a0:       pxor   %xmm0,%xmm0
 0.00 :        832a4:       sub    $0x1000,%r10
 0.00 :        832ab:       jmpq   831fa <__GI_strcmp+0xb9a>
 0.00 :        832b0:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :        832b5:       psrldq $0x9,%xmm0
 0.00 :        832ba:       psrldq $0x9,%xmm3
 0.00 :        832bf:       jmpq   83a40 <__GI_strcmp+0x13e0>
 0.00 :        832c4:       data32 data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :        832d0:       pxor   %xmm0,%xmm0
 0.00 :        832d4:       movdqa (%rdi),%xmm2
 0.00 :        832d8:       movdqa (%rsi),%xmm1
 0.00 :        832dc:       pcmpeqb %xmm1,%xmm0
 0.00 :        832e0:       pslldq $0x6,%xmm2
 0.00 :        832e5:       pcmpeqb %xmm1,%xmm2
 0.00 :        832e9:       psubb  %xmm0,%xmm2
 0.00 :        832ed:       pmovmskb %xmm2,%r9d
 0.00 :        832f2:       shr    %cl,%edx
 0.00 :        832f4:       shr    %cl,%r9d
 0.00 :        832f7:       sub    %r9d,%edx
 0.00 :        832fa:       jne    83a55 <__GI_strcmp+0x13f5>
 0.00 :        83300:       movdqa (%rdi),%xmm3
 0.00 :        83304:       pxor   %xmm0,%xmm0
 0.00 :        83308:       mov    $0x10,%rcx
 0.00 :        8330f:       mov    $0xa,%r9d
 0.00 :        83315:       lea    0xa(%rdi),%r10
 0.00 :        83319:       and    $0xfff,%r10
 0.00 :        83320:       sub    $0x1000,%r10
 0.00 :        83327:       nopw   0x0(%rax,%rax,1)
 0.00 :        83330:       add    $0x10,%r10
 0.00 :        83334:       jg     833d0 <__GI_strcmp+0xd70>
 0.00 :        8333a:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :        8333f:       movdqa (%rdi,%rcx,1),%xmm2
 0.00 :        83344:       movdqa %xmm2,%xmm4
 0.00 :        83348:       psrldq $0xa,%xmm3
 0.00 :        8334d:       pslldq $0x6,%xmm2
 0.00 :        83352:       por    %xmm3,%xmm2
 0.00 :        83356:       pcmpeqb %xmm1,%xmm0
 0.00 :        8335a:       pcmpeqb %xmm2,%xmm1
 0.00 :        8335e:       psubb  %xmm0,%xmm1
 0.00 :        83362:       pmovmskb %xmm1,%edx
 0.00 :        83366:       sub    $0xffff,%edx
 0.00 :        8336c:       jne    83a50 <__GI_strcmp+0x13f0>
 0.00 :        83372:       add    $0x10,%rcx
 0.00 :        83376:       movdqa %xmm4,%xmm3
 0.00 :        8337a:       add    $0x10,%r10
 0.00 :        8337e:       jg     833d0 <__GI_strcmp+0xd70>
 0.00 :        83380:       movdqa (%rsi,%rcx,1),%xmm1
```

```
 0.00 :        83385:       movdqa (%rdi,%rcx,1),%xmm2
 0.00 :        8338a:       movdqa %xmm2,%xmm4
 0.00 :        8338e:       psrldq $0xa,%xmm3
 0.00 :        83393:       pslldq $0x6,%xmm2
 0.00 :        83398:       por    %xmm3,%xmm2
 0.00 :        8339c:       pcmpeqb %xmm1,%xmm0
 0.00 :        833a0:       pcmpeqb %xmm2,%xmm1
 0.00 :        833a4:       psubb  %xmm0,%xmm1
 0.00 :        833a8:       pmovmskb %xmm1,%edx
 0.00 :        833ac:       sub    $0xffff,%edx
 0.00 :        833b2:       jne    83a50 <__GI_strcmp+0x13f0>
 0.00 :        833b8:       add    $0x10,%rcx
 0.00 :        833bc:       movdqa %xmm4,%xmm3
 0.00 :        833c0:       jmpq   83330 <__GI_strcmp+0xcd0>
 0.00 :        833c5:       data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :        833d0:       pcmpeqb %xmm3,%xmm0
 0.00 :        833d4:       pmovmskb %xmm0,%edx
 0.00 :        833d8:       test   $0xfc00,%edx
 0.00 :        833de:       jne    833f0 <__GI_strcmp+0xd90>
 0.00 :        833e0:       pxor   %xmm0,%xmm0
 0.00 :        833e4:       sub    $0x1000,%r10
 0.00 :        833eb:       jmpq   8333a <__GI_strcmp+0xcda>
 0.00 :        833f0:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :        833f5:       psrldq $0xa,%xmm0
 0.00 :        833fa:       psrldq $0xa,%xmm3
 0.00 :        833ff:       jmpq   83a40 <__GI_strcmp+0x13e0>
 0.00 :        83404:       data32 data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :        83410:       pxor   %xmm0,%xmm0
 0.00 :        83414:       movdqa (%rdi),%xmm2
 0.00 :        83418:       movdqa (%rsi),%xmm1
 0.00 :        8341c:       pcmpeqb %xmm1,%xmm0
 0.00 :        83420:       pslldq $0x5,%xmm2
 0.00 :        83425:       pcmpeqb %xmm1,%xmm2
 0.00 :        83429:       psubb  %xmm0,%xmm2
 0.00 :        8342d:       pmovmskb %xmm2,%r9d
 0.00 :        83432:       shr    %cl,%edx
 0.00 :        83434:       shr    %cl,%r9d
 0.00 :        83437:       sub    %r9d,%edx
 0.00 :        8343a:       jne    83a55 <__GI_strcmp+0x13f5>
 0.00 :        83440:       movdqa (%rdi),%xmm3
 0.00 :        83444:       pxor   %xmm0,%xmm0
 0.00 :        83448:       mov    $0x10,%rcx
 0.00 :        8344f:       mov    $0xb,%r9d
 0.00 :        83455:       lea    0xb(%rdi),%r10
 0.00 :        83459:       and    $0xfff,%r10
 0.00 :        83460:       sub    $0x1000,%r10
 0.00 :        83467:       nopw   0x0(%rax,%rax,1)
 0.00 :        83470:       add    $0x10,%r10
 0.00 :        83474:       jg     83510 <__GI_strcmp+0xeb0>
 0.00 :        8347a:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :        8347f:       movdqa (%rdi,%rcx,1),%xmm2
 0.00 :        83484:       movdqa %xmm2,%xmm4
 0.00 :        83488:       psrldq $0xb,%xmm3
 0.00 :        8348d:       pslldq $0x5,%xmm2
 0.00 :        83492:       por    %xmm3,%xmm2
 0.00 :        83496:       pcmpeqb %xmm1,%xmm0
 0.00 :        8349a:       pcmpeqb %xmm2,%xmm1
 0.00 :        8349e:       psubb  %xmm0,%xmm1
 0.00 :        834a2:       pmovmskb %xmm1,%edx
 0.00 :        834a6:       sub    $0xffff,%edx
 0.00 :        834ac:       jne    83a50 <__GI_strcmp+0x13f0>
 0.00 :        834b2:       add    $0x10,%rcx
 0.00 :        834b6:       movdqa %xmm4,%xmm3
 0.00 :        834ba:       add    $0x10,%r10
```

```
 0.00 :         834be:       jg     83510 <__GI_strcmp+0xeb0>
 0.00 :         834c0:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :         834c5:       movdqa (%rdi,%rcx,1),%xmm2
 0.00 :         834ca:       movdqa %xmm2,%xmm4
 0.00 :         834ce:       psrldq $0xb,%xmm3
 0.00 :         834d3:       pslldq $0x5,%xmm2
 0.00 :         834d8:       por    %xmm3,%xmm2
 0.00 :         834dc:       pcmpeqb %xmm1,%xmm0
 0.00 :         834e0:       pcmpeqb %xmm2,%xmm1
 0.00 :         834e4:       psubb  %xmm0,%xmm1
 0.00 :         834e8:       pmovmskb %xmm1,%edx
 0.00 :         834ec:       sub    $0xffff,%edx
 0.00 :         834f2:       jne    83a50 <__GI_strcmp+0x13f0>
 0.00 :         834f8:       add    $0x10,%rcx
 0.00 :         834fc:       movdqa %xmm4,%xmm3
 0.00 :         83500:       jmpq   83470 <__GI_strcmp+0xe10>
 0.00 :         83505:       data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :         83510:       pcmpeqb %xmm3,%xmm0
 0.00 :         83514:       pmovmskb %xmm0,%edx
 0.00 :         83518:       test   $0xf800,%edx
 0.00 :         8351e:       jne    83530 <__GI_strcmp+0xed0>
 0.00 :         83520:       pxor   %xmm0,%xmm0
 0.00 :         83524:       sub    $0x1000,%r10
 0.00 :         8352b:       jmpq   8347a <__GI_strcmp+0xe1a>
 0.00 :         83530:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :         83535:       psrldq $0xb,%xmm0
 0.00 :         8353a:       psrldq $0xb,%xmm3
 0.00 :         8353f:       jmpq   83a40 <__GI_strcmp+0x13e0>
 0.00 :         83544:       data32 data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :         83550:       pxor   %xmm0,%xmm0
 0.00 :         83554:       movdqa (%rdi),%xmm2
 0.00 :         83558:       movdqa (%rsi),%xmm1
 0.00 :         8355c:       pcmpeqb %xmm1,%xmm0
 0.00 :         83560:       pslldq $0x4,%xmm2
 0.00 :         83565:       pcmpeqb %xmm1,%xmm2
 0.00 :         83569:       psubb  %xmm0,%xmm2
 0.00 :         8356d:       pmovmskb %xmm2,%r9d
 0.00 :         83572:       shr    %cl,%edx
 0.00 :         83574:       shr    %cl,%r9d
 0.00 :         83577:       sub    %r9d,%edx
 0.00 :         8357a:       jne    83a55 <__GI_strcmp+0x13f5>
 0.00 :         83580:       movdqa (%rdi),%xmm3
 0.00 :         83584:       pxor   %xmm0,%xmm0
 0.00 :         83588:       mov    $0x10,%rcx
 0.00 :         8358f:       mov    $0xc,%r9d
 0.00 :         83595:       lea    0xc(%rdi),%r10
 0.00 :         83599:       and    $0xfff,%r10
 0.00 :         835a0:       sub    $0x1000,%r10
 0.00 :         835a7:       nopw   0x0(%rax,%rax,1)
 0.00 :         835b0:       add    $0x10,%r10
 0.00 :         835b4:       jg     83650 <__GI_strcmp+0xff0>
 0.00 :         835ba:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :         835bf:       movdqa (%rdi,%rcx,1),%xmm2
 0.00 :         835c4:       movdqa %xmm2,%xmm4
 0.00 :         835c8:       psrldq $0xc,%xmm3
 0.00 :         835cd:       pslldq $0x4,%xmm2
 0.00 :         835d2:       por    %xmm3,%xmm2
 0.00 :         835d6:       pcmpeqb %xmm1,%xmm0
 0.00 :         835da:       pcmpeqb %xmm2,%xmm1
 0.00 :         835de:       psubb  %xmm0,%xmm1
 0.00 :         835e2:       pmovmskb %xmm1,%edx
 0.00 :         835e6:       sub    $0xffff,%edx
 0.00 :         835ec:       jne    83a50 <__GI_strcmp+0x13f0>
 0.00 :         835f2:       add    $0x10,%rcx
```

```
 0.00 :         835f6:       movdqa %xmm4,%xmm3
 0.00 :         835fa:       add    $0x10,%r10
 0.00 :         835fe:       jg     83650 <__GI_strcmp+0xff0>
 0.00 :         83600:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :         83605:       movdqa (%rdi,%rcx,1),%xmm2
 0.00 :         8360a:       movdqa %xmm2,%xmm4
 0.00 :         8360e:       psrldq $0xc,%xmm3
 0.00 :         83613:       pslldq $0x4,%xmm2
 0.00 :         83618:       por    %xmm3,%xmm2
 0.00 :         8361c:       pcmpeqb %xmm1,%xmm0
 0.00 :         83620:       pcmpeqb %xmm2,%xmm1
 0.00 :         83624:       psubb  %xmm0,%xmm1
 0.00 :         83628:       pmovmskb %xmm1,%edx
 0.00 :         8362c:       sub    $0xffff,%edx
 0.00 :         83632:       jne    83a50 <__GI_strcmp+0x13f0>
 0.00 :         83638:       add    $0x10,%rcx
 0.00 :         8363c:       movdqa %xmm4,%xmm3
 0.00 :         83640:       jmpq   835b0 <__GI_strcmp+0xf50>
 0.00 :         83645:       data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :         83650:       pcmpeqb %xmm3,%xmm0
 0.00 :         83654:       pmovmskb %xmm0,%edx
 0.00 :         83658:       test   $0xf000,%edx
 0.00 :         8365e:       jne    83670 <__GI_strcmp+0x1010>
 0.00 :         83660:       pxor   %xmm0,%xmm0
 0.00 :         83664:       sub    $0x1000,%r10
 0.00 :         8366b:       jmpq   835ba <__GI_strcmp+0xf5a>
 0.00 :         83670:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :         83675:       psrldq $0xc,%xmm0
 0.00 :         8367a:       psrldq $0xc,%xmm3
 0.00 :         8367f:       jmpq   83a40 <__GI_strcmp+0x13e0>
 0.00 :         83684:       data32 data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :         83690:       pxor   %xmm0,%xmm0
 0.00 :         83694:       movdqa (%rdi),%xmm2
 0.00 :         83698:       movdqa (%rsi),%xmm1
 0.00 :         8369c:       pcmpeqb %xmm1,%xmm0
 0.00 :         836a0:       pslldq $0x3,%xmm2
 0.00 :         836a5:       pcmpeqb %xmm1,%xmm2
 0.00 :         836a9:       psubb  %xmm0,%xmm2
 0.00 :         836ad:       pmovmskb %xmm2,%r9d
 0.00 :         836b2:       shr    %cl,%edx
 0.00 :         836b4:       shr    %cl,%r9d
 0.00 :         836b7:       sub    %r9d,%edx
 0.00 :         836ba:       jne    83a55 <__GI_strcmp+0x13f5>
 0.00 :         836c0:       movdqa (%rdi),%xmm3
 0.00 :         836c4:       pxor   %xmm0,%xmm0
 0.00 :         836c8:       mov    $0x10,%rcx
 0.00 :         836cf:       mov    $0xd,%r9d
 0.00 :         836d5:       lea    0xd(%rdi),%r10
 0.00 :         836d9:       and    $0xfff,%r10
 0.00 :         836e0:       sub    $0x1000,%r10
 0.00 :         836e7:       nopw   0x0(%rax,%rax,1)
 0.00 :         836f0:       add    $0x10,%r10
 0.00 :         836f4:       jg     83790 <__GI_strcmp+0x1130>
 0.00 :         836fa:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :         836ff:       movdqa (%rdi,%rcx,1),%xmm2
 0.00 :         83704:       movdqa %xmm2,%xmm4
 0.00 :         83708:       psrldq $0xd,%xmm3
 0.00 :         8370d:       pslldq $0x3,%xmm2
 0.00 :         83712:       por    %xmm3,%xmm2
 0.00 :         83716:       pcmpeqb %xmm1,%xmm0
 0.00 :         8371a:       pcmpeqb %xmm2,%xmm1
 0.00 :         8371e:       psubb  %xmm0,%xmm1
 0.00 :         83722:       pmovmskb %xmm1,%edx
 0.00 :         83726:       sub    $0xffff,%edx
```

```
 0.00 :        8372c:       jne    83a50 <__GI_strcmp+0x13f0>
 0.00 :        83732:       add    $0x10,%rcx
 0.00 :        83736:       movdqa %xmm4,%xmm3
 0.00 :        8373a:       add    $0x10,%r10
 0.00 :        8373e:       jg     83790 <__GI_strcmp+0x1130>
 0.00 :        83740:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :        83745:       movdqa (%rdi,%rcx,1),%xmm2
 0.00 :        8374a:       movdqa %xmm2,%xmm4
 0.00 :        8374e:       psrldq $0xd,%xmm3
 0.00 :        83753:       pslldq $0x3,%xmm2
 0.00 :        83758:       por    %xmm3,%xmm2
 0.00 :        8375c:       pcmpeqb %xmm1,%xmm0
 0.00 :        83760:       pcmpeqb %xmm2,%xmm1
 0.00 :        83764:       psubb  %xmm0,%xmm1
 0.00 :        83768:       pmovmskb %xmm1,%edx
 0.00 :        8376c:       sub    $0xffff,%edx
 0.00 :        83772:       jne    83a50 <__GI_strcmp+0x13f0>
 0.00 :        83778:       add    $0x10,%rcx
 0.00 :        8377c:       movdqa %xmm4,%xmm3
 0.00 :        83780:       jmpq   836f0 <__GI_strcmp+0x1090>
 0.00 :        83785:       data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :        83790:       pcmpeqb %xmm3,%xmm0
 0.00 :        83794:       pmovmskb %xmm0,%edx
 0.00 :        83798:       test   $0xe000,%edx
 0.00 :        8379e:       jne    837b0 <__GI_strcmp+0x1150>
 0.00 :        837a0:       pxor   %xmm0,%xmm0
 0.00 :        837a4:       sub    $0x1000,%r10
 0.00 :        837ab:       jmpq   836fa <__GI_strcmp+0x109a>
 0.00 :        837b0:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :        837b5:       psrldq $0xd,%xmm0
 0.00 :        837ba:       psrldq $0xd,%xmm3
 0.00 :        837bf:       jmpq   83a40 <__GI_strcmp+0x13e0>
 0.00 :        837c4:       data32 data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :        837d0:       pxor   %xmm0,%xmm0
 0.00 :        837d4:       movdqa (%rdi),%xmm2
 0.00 :        837d8:       movdqa (%rsi),%xmm1
 0.00 :        837dc:       pcmpeqb %xmm1,%xmm0
 0.00 :        837e0:       pslldq $0x2,%xmm2
 0.00 :        837e5:       pcmpeqb %xmm1,%xmm2
 0.00 :        837e9:       psubb  %xmm0,%xmm2
 0.00 :        837ed:       pmovmskb %xmm2,%r9d
 0.00 :        837f2:       shr    %cl,%edx
 0.00 :        837f4:       shr    %cl,%r9d
 0.00 :        837f7:       sub    %r9d,%edx
 0.00 :        837fa:       jne    83a55 <__GI_strcmp+0x13f5>
 0.00 :        83800:       movdqa (%rdi),%xmm3
 0.00 :        83804:       pxor   %xmm0,%xmm0
 0.00 :        83808:       mov    $0x10,%rcx
 0.00 :        8380f:       mov    $0xe,%r9d
 0.00 :        83815:       lea    0xe(%rdi),%r10
 0.00 :        83819:       and    $0xfff,%r10
 0.00 :        83820:       sub    $0x1000,%r10
 0.00 :        83827:       nopw   0x0(%rax,%rax,1)
 0.00 :        83830:       add    $0x10,%r10
 0.00 :        83834:       jg     838d0 <__GI_strcmp+0x1270>
 0.00 :        8383a:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :        8383f:       movdqa (%rdi,%rcx,1),%xmm2
 0.00 :        83844:       movdqa %xmm2,%xmm4
 0.00 :        83848:       psrldq $0xe,%xmm3
 0.00 :        8384d:       pslldq $0x2,%xmm2
 0.00 :        83852:       por    %xmm3,%xmm2
 0.00 :        83856:       pcmpeqb %xmm1,%xmm0
 0.00 :        8385a:       pcmpeqb %xmm2,%xmm1
 0.00 :        8385e:       psubb  %xmm0,%xmm1
```

```
 0.00 :        83862:       pmovmskb %xmm1,%edx
 0.00 :        83866:       sub    $0xffff,%edx
 0.00 :        8386c:       jne    83a50 <__GI_strcmp+0x13f0>
 0.00 :        83872:       add    $0x10,%rcx
 0.00 :        83876:       movdqa %xmm4,%xmm3
 0.00 :        8387a:       add    $0x10,%r10
 0.00 :        8387e:       jg     838d0 <__GI_strcmp+0x1270>
 0.00 :        83880:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :        83885:       movdqa (%rdi,%rcx,1),%xmm2
 0.00 :        8388a:       movdqa %xmm2,%xmm4
 0.00 :        8388e:       psrldq $0xe,%xmm3
 0.00 :        83893:       pslldq $0x2,%xmm2
 0.00 :        83898:       por    %xmm3,%xmm2
 0.00 :        8389c:       pcmpeqb %xmm1,%xmm0
 0.00 :        838a0:       pcmpeqb %xmm2,%xmm1
 0.00 :        838a4:       psubb  %xmm0,%xmm1
 0.00 :        838a8:       pmovmskb %xmm1,%edx
 0.00 :        838ac:       sub    $0xffff,%edx
 0.00 :        838b2:       jne    83a50 <__GI_strcmp+0x13f0>
 0.00 :        838b8:       add    $0x10,%rcx
 0.00 :        838bc:       movdqa %xmm4,%xmm3
 0.00 :        838c0:       jmpq   83830 <__GI_strcmp+0x11d0>
 0.00 :        838c5:       data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :        838d0:       pcmpeqb %xmm3,%xmm0
 0.00 :        838d4:       pmovmskb %xmm0,%edx
 0.00 :        838d8:       test   $0xc000,%edx
 0.00 :        838de:       jne    838f0 <__GI_strcmp+0x1290>
 0.00 :        838e0:       pxor   %xmm0,%xmm0
 0.00 :        838e4:       sub    $0x1000,%r10
 0.00 :        838eb:       jmpq   8383a <__GI_strcmp+0x11da>
 0.00 :        838f0:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :        838f5:       psrldq $0xe,%xmm0
 0.00 :        838fa:       psrldq $0xe,%xmm3
 0.00 :        838ff:       jmpq   83a40 <__GI_strcmp+0x13e0>
 0.00 :        83904:       data32 data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :        83910:       pxor   %xmm0,%xmm0
 0.00 :        83914:       movdqa (%rdi),%xmm2
 0.00 :        83918:       movdqa (%rsi),%xmm1
 0.00 :        8391c:       pcmpeqb %xmm1,%xmm0
 0.00 :        83920:       pslldq $0x1,%xmm2
 0.00 :        83925:       pcmpeqb %xmm1,%xmm2
 0.00 :        83929:       psubb  %xmm0,%xmm2
 0.00 :        8392d:       pmovmskb %xmm2,%r9d
 0.00 :        83932:       shr    %cl,%edx
 0.00 :        83934:       shr    %cl,%r9d
 0.00 :        83937:       sub    %r9d,%edx
 0.00 :        8393a:       jne    83a55 <__GI_strcmp+0x13f5>
 0.00 :        83940:       movdqa (%rdi),%xmm3
 0.00 :        83944:       pxor   %xmm0,%xmm0
 0.00 :        83948:       mov    $0x10,%rcx
 0.00 :        8394f:       mov    $0xf,%r9d
 0.00 :        83955:       lea    0xf(%rdi),%r10
 0.00 :        83959:       and    $0xfff,%r10
 0.00 :        83960:       sub    $0x1000,%r10
 0.00 :        83967:       nopw   0x0(%rax,%rax,1)
 0.00 :        83970:       add    $0x10,%r10
 0.00 :        83974:       jg     83a10 <__GI_strcmp+0x13b0>
 0.00 :        8397a:       movdqa (%rsi,%rcx,1),%xmm1
 0.00 :        8397f:       movdqa (%rdi,%rcx,1),%xmm2
 0.00 :        83984:       movdqa %xmm2,%xmm4
 0.00 :        83988:       psrldq $0xf,%xmm3
 0.00 :        8398d:       pslldq $0x1,%xmm2
 0.00 :        83992:       por    %xmm3,%xmm2
 0.00 :        83996:       pcmpeqb %xmm1,%xmm0
```

```
    0.00 :          8399a:       pcmpeqb %xmm2,%xmm1
    0.00 :          8399e:       psubb  %xmm0,%xmm1
    0.00 :          839a2:       pmovmskb %xmm1,%edx
    0.00 :          839a6:       sub    $0xffff,%edx
    0.00 :          839ac:       jne    83a50 <__GI_strcmp+0x13f0>
    0.00 :          839b2:       add    $0x10,%rcx
    0.00 :          839b6:       movdqa %xmm4,%xmm3
    0.00 :          839ba:       add    $0x10,%r10
    0.00 :          839be:       jg     83a10 <__GI_strcmp+0x13b0>
    0.00 :          839c0:       movdqa (%rsi,%rcx,1),%xmm1
    0.00 :          839c5:       movdqa (%rdi,%rcx,1),%xmm2
    0.00 :          839ca:       movdqa %xmm2,%xmm4
    0.00 :          839ce:       psrldq $0xf,%xmm3
    0.00 :          839d3:       pslldq $0x1,%xmm2
    0.00 :          839d8:       por    %xmm3,%xmm2
    0.00 :          839dc:       pcmpeqb %xmm1,%xmm0
    0.00 :          839e0:       pcmpeqb %xmm2,%xmm1
    0.00 :          839e4:       psubb  %xmm0,%xmm1
    0.00 :          839e8:       pmovmskb %xmm1,%edx
    0.00 :          839ec:       sub    $0xffff,%edx
    0.00 :          839f2:       jne    83a50 <__GI_strcmp+0x13f0>
    0.00 :          839f4:       add    $0x10,%rcx
    0.00 :          839f8:       movdqa %xmm4,%xmm3
    0.00 :          839fc:       jmpq   83970 <__GI_strcmp+0x1310>
    0.00 :          83a01:       data32 data32 data32 data32 data32 nopw %cs:0x0(%rax,%rax,1)
    0.00 :          83a10:       pcmpeqb %xmm3,%xmm0
    0.00 :          83a14:       pmovmskb %xmm0,%edx
    0.00 :          83a18:       test   $0x8000,%edx
    0.00 :          83a1e:       jne    83a30 <__GI_strcmp+0x13d0>
    0.00 :          83a20:       pxor   %xmm0,%xmm0
    0.00 :          83a24:       sub    $0x1000,%r10
    0.00 :          83a2b:       jmpq   8397a <__GI_strcmp+0x131a>
    0.00 :          83a30:       movdqa (%rsi,%rcx,1),%xmm1
    0.00 :          83a35:       psrldq $0xf,%xmm3
    0.00 :          83a3a:       psrldq $0xf,%xmm0
    0.00 :          83a3f:       nop
    0.00 :          83a40:       pcmpeqb %xmm3,%xmm1
    0.00 :          83a44:       psubb  %xmm0,%xmm1
    0.00 :          83a48:       pmovmskb %xmm1,%edx
    0.00 :          83a4c:       not    %edx
    0.00 :          83a4e:       xchg   %ax,%ax
    0.00 :          83a50:       lea    -0x10(%r9,%rcx,1),%rax
    0.00 :          83a55:       lea    (%rdi,%rax,1),%rdi
    0.00 :          83a59:       lea    (%rsi,%rcx,1),%rsi
    0.00 :          83a5d:       test   %r8d,%r8d
    0.00 :          83a60:       je     83a70 <__GI_strcmp+0x1410>
    0.00 :          83a62:       xchg   %rsi,%rdi
    0.00 :          83a65:       data32 nopw %cs:0x0(%rax,%rax,1)
    0.00 :          83a70:       bsf    %rdx,%rdx
    0.00 :          83a74:       movzbl (%rsi,%rdx,1),%ecx
    0.00 :          83a78:       movzbl (%rdi,%rdx,1),%eax
    0.00 :          83a7c:       sub    %ecx,%eax
  100.00 :          83a7e:       retq
    0.00 :          83a7f:       xor    %eax,%eax
    0.00 :          83a81:       retq
    0.00 :          83a82:       data32 data32 data32 data32 nopw %cs:0x0(%rax,%rax,1)
    0.00 :          83a90:       movzbl (%rsi),%ecx
    0.00 :          83a93:       movzbl (%rdi),%eax
    0.00 :          83a96:       sub    %ecx,%eax
    0.00 :          83a98:       retq
 Percent |       Source code & Disassembly of libc-2.17.so
-----------------------------------------------
         :
         :
```

```
       :
       :        Disassembly of section .text:
       :
       :        0000000000086cd0 <__GI_memset>:
  0.00 :          86cd0:       cmp    $0x1,%rdx
  0.00 :          86cd4:       mov    %rdi,%rax
  0.00 :          86cd7:       jne    86cdd <__GI_memset+0xd>
  0.00 :          86cd9:       mov    %sil,(%rdi)
  0.00 :          86cdc:       retq
  0.00 :          86cdd:       movabs $0x101010101010101,%r9
  0.00 :          86ce7:       mov    %rdx,%r8
  0.00 :          86cea:       movzbq %sil,%rdx
  0.00 :          86cee:       imul   %r9,%rdx
  0.00 :          86cf2:       cmp    $0x90,%r8
  0.00 :          86cf9:       ja     86fe0 <__GI_memset+0x310>
  0.00 :          86cff:       add    %r8,%rdi
  0.00 :          86d02:       lea    0x13(%rip),%r11        # 86d1c <__GI_memset+0x4c>
  0.00 :          86d09:       lea    0xdc380(%rip),%rcx        # 163090 <result_type.7642+0x70>
  0.00 :          86d10:       movswq (%rcx,%r8,2),%rcx
  0.00 :          86d15:       lea    (%rcx,%r11,1),%r11
  0.00 :          86d19:       jmpq   *%r11
  0.00 :          86d1c:       retq
  0.00 :          86d1d:       nopl   (%rax)
  0.00 :          86d20:       mov    %rdx,-0x89(%rdi)
  0.00 :          86d27:       mov    %rdx,-0x81(%rdi)
  0.00 :          86d2e:       mov    %rdx,-0x79(%rdi)
  0.00 :          86d32:       mov    %rdx,-0x71(%rdi)
  0.00 :          86d36:       mov    %rdx,-0x69(%rdi)
  0.00 :          86d3a:       mov    %rdx,-0x61(%rdi)
  0.00 :          86d3e:       mov    %rdx,-0x59(%rdi)
  0.00 :          86d42:       mov    %rdx,-0x51(%rdi)
  0.00 :          86d46:       mov    %rdx,-0x49(%rdi)
  0.00 :          86d4a:       mov    %rdx,-0x41(%rdi)
  0.00 :          86d4e:       mov    %rdx,-0x39(%rdi)
  0.00 :          86d52:       mov    %rdx,-0x31(%rdi)
  0.00 :          86d56:       mov    %rdx,-0x29(%rdi)
  0.00 :          86d5a:       mov    %rdx,-0x21(%rdi)
  0.00 :          86d5e:       mov    %rdx,-0x19(%rdi)
  0.00 :          86d62:       mov    %rdx,-0x11(%rdi)
  0.00 :          86d66:       mov    %rdx,-0x9(%rdi)
  0.00 :          86d6a:       mov    %dl,-0x1(%rdi)
  0.00 :          86d6d:       retq
  0.00 :          86d6e:       xchg   %ax,%ax
  0.00 :          86d70:       mov    %rdx,-0x90(%rdi)
  0.00 :          86d77:       mov    %rdx,-0x88(%rdi)
  0.00 :          86d7e:       mov    %rdx,-0x80(%rdi)
  0.00 :          86d82:       mov    %rdx,-0x78(%rdi)
  0.00 :          86d86:       mov    %rdx,-0x70(%rdi)
  0.00 :          86d8a:       mov    %rdx,-0x68(%rdi)
  0.00 :          86d8e:       mov    %rdx,-0x60(%rdi)
  0.00 :          86d92:       mov    %rdx,-0x58(%rdi)
  0.00 :          86d96:       mov    %rdx,-0x50(%rdi)
  0.00 :          86d9a:       mov    %rdx,-0x48(%rdi)
  0.00 :          86d9e:       mov    %rdx,-0x40(%rdi)
  0.00 :          86da2:       mov    %rdx,-0x38(%rdi)
  0.00 :          86da6:       mov    %rdx,-0x30(%rdi)
  0.00 :          86daa:       mov    %rdx,-0x28(%rdi)
  0.00 :          86dae:       mov    %rdx,-0x20(%rdi)
  0.00 :          86db2:       mov    %rdx,-0x18(%rdi)
  0.00 :          86db6:       mov    %rdx,-0x10(%rdi)
  0.00 :          86dba:       mov    %rdx,-0x8(%rdi)
  0.00 :          86dbe:       retq
  0.00 :          86dbf:       nop
  0.00 :          86dc0:       mov    %rdx,-0x8a(%rdi)
```

```
 0.00 :         86dc7:       mov    %rdx,-0x82(%rdi)
 0.00 :         86dce:       mov    %rdx,-0x7a(%rdi)
 0.00 :         86dd2:       mov    %rdx,-0x72(%rdi)
 0.00 :         86dd6:       mov    %rdx,-0x6a(%rdi)
 0.00 :         86dda:       mov    %rdx,-0x62(%rdi)
 0.00 :         86dde:       mov    %rdx,-0x5a(%rdi)
 0.00 :         86de2:       mov    %rdx,-0x52(%rdi)
 0.00 :         86de6:       mov    %rdx,-0x4a(%rdi)
 0.00 :         86dea:       mov    %rdx,-0x42(%rdi)
 0.00 :         86dee:       mov    %rdx,-0x3a(%rdi)
 0.00 :         86df2:       mov    %rdx,-0x32(%rdi)
 0.00 :         86df6:       mov    %rdx,-0x2a(%rdi)
 0.00 :         86dfa:       mov    %rdx,-0x22(%rdi)
 0.00 :         86dfe:       mov    %rdx,-0x1a(%rdi)
 0.00 :         86e02:       mov    %rdx,-0x12(%rdi)
 0.00 :         86e06:       mov    %rdx,-0xa(%rdi)
 0.00 :         86e0a:       mov    %dx,-0x2(%rdi)
 0.00 :         86e0e:       retq
 0.00 :         86e0f:       nop
 0.00 :         86e10:       mov    %rdx,-0x8b(%rdi)
 0.00 :         86e17:       mov    %rdx,-0x83(%rdi)
 0.00 :         86e1e:       mov    %rdx,-0x7b(%rdi)
 0.00 :         86e22:       mov    %rdx,-0x73(%rdi)
 0.00 :         86e26:       mov    %rdx,-0x6b(%rdi)
 0.00 :         86e2a:       mov    %rdx,-0x63(%rdi)
 0.00 :         86e2e:       mov    %rdx,-0x5b(%rdi)
 0.00 :         86e32:       mov    %rdx,-0x53(%rdi)
 0.00 :         86e36:       mov    %rdx,-0x4b(%rdi)
 0.00 :         86e3a:       mov    %rdx,-0x43(%rdi)
 0.00 :         86e3e:       mov    %rdx,-0x3b(%rdi)
 0.00 :         86e42:       mov    %rdx,-0x33(%rdi)
 0.00 :         86e46:       mov    %rdx,-0x2b(%rdi)
 0.00 :         86e4a:       mov    %rdx,-0x23(%rdi)
 0.00 :         86e4e:       mov    %rdx,-0x1b(%rdi)
 0.00 :         86e52:       mov    %rdx,-0x13(%rdi)
 0.00 :         86e56:       mov    %rdx,-0xb(%rdi)
 0.00 :         86e5a:       mov    %dx,-0x3(%rdi)
 0.00 :         86e5e:       mov    %dl,-0x1(%rdi)
 0.00 :         86e61:       retq
 0.00 :         86e62:       data32 data32 data32 data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :         86e70:       mov    %rdx,-0x8c(%rdi)
 0.00 :         86e77:       mov    %rdx,-0x84(%rdi)
 0.00 :         86e7e:       mov    %rdx,-0x7c(%rdi)
 0.00 :         86e82:       mov    %rdx,-0x74(%rdi)
 0.00 :         86e86:       mov    %rdx,-0x6c(%rdi)
 0.00 :         86e8a:       mov    %rdx,-0x64(%rdi)
 0.00 :         86e8e:       mov    %rdx,-0x5c(%rdi)
 0.00 :         86e92:       mov    %rdx,-0x54(%rdi)
 0.00 :         86e96:       mov    %rdx,-0x4c(%rdi)
 0.00 :         86e9a:       mov    %rdx,-0x44(%rdi)
 0.00 :         86e9e:       mov    %rdx,-0x3c(%rdi)
 0.00 :         86ea2:       mov    %rdx,-0x34(%rdi)
 0.00 :         86ea6:       mov    %rdx,-0x2c(%rdi)
 0.00 :         86eaa:       mov    %rdx,-0x24(%rdi)
 0.00 :         86eae:       mov    %rdx,-0x1c(%rdi)
 0.00 :         86eb2:       mov    %rdx,-0x14(%rdi)
 0.00 :         86eb6:       mov    %rdx,-0xc(%rdi)
 0.00 :         86eba:       mov    %edx,-0x4(%rdi)
 0.00 :         86ebd:       retq
 0.00 :         86ebe:       xchg   %ax,%ax
 0.00 :         86ec0:       mov    %rdx,-0x8d(%rdi)
 0.00 :         86ec7:       mov    %rdx,-0x85(%rdi)
 0.00 :         86ece:       mov    %rdx,-0x7d(%rdi)
 0.00 :         86ed2:       mov    %rdx,-0x75(%rdi)
```

```
 0.00 :         86ed6:       mov    %rdx,-0x6d(%rdi)
 0.00 :         86eda:       mov    %rdx,-0x65(%rdi)
 0.00 :         86ede:       mov    %rdx,-0x5d(%rdi)
 0.00 :         86ee2:       mov    %rdx,-0x55(%rdi)
 0.00 :         86ee6:       mov    %rdx,-0x4d(%rdi)
 0.00 :         86eea:       mov    %rdx,-0x45(%rdi)
 0.00 :         86eee:       mov    %rdx,-0x3d(%rdi)
 0.00 :         86ef2:       mov    %rdx,-0x35(%rdi)
 0.00 :         86ef6:       mov    %rdx,-0x2d(%rdi)
 0.00 :         86efa:       mov    %rdx,-0x25(%rdi)
 0.00 :         86efe:       mov    %rdx,-0x1d(%rdi)
 0.00 :         86f02:       mov    %rdx,-0x15(%rdi)
 0.00 :         86f06:       mov    %rdx,-0xd(%rdi)
 0.00 :         86f0a:       mov    %edx,-0x5(%rdi)
 0.00 :         86f0d:       mov    %dl,-0x1(%rdi)
 0.00 :         86f10:       retq
 0.00 :         86f11:       data32 data32 data32 data32 data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :         86f20:       mov    %rdx,-0x8e(%rdi)
 0.00 :         86f27:       mov    %rdx,-0x86(%rdi)
 0.00 :         86f2e:       mov    %rdx,-0x7e(%rdi)
 0.00 :         86f32:       mov    %rdx,-0x76(%rdi)
 0.00 :         86f36:       mov    %rdx,-0x6e(%rdi)
 0.00 :         86f3a:       mov    %rdx,-0x66(%rdi)
 0.00 :         86f3e:       mov    %rdx,-0x5e(%rdi)
 0.00 :         86f42:       mov    %rdx,-0x56(%rdi)
 0.00 :         86f46:       mov    %rdx,-0x4e(%rdi)
 0.00 :         86f4a:       mov    %rdx,-0x46(%rdi)
 0.00 :         86f4e:       mov    %rdx,-0x3e(%rdi)
 0.00 :         86f52:       mov    %rdx,-0x36(%rdi)
 0.00 :         86f56:       mov    %rdx,-0x2e(%rdi)
 0.00 :         86f5a:       mov    %rdx,-0x26(%rdi)
 0.00 :         86f5e:       mov    %rdx,-0x1e(%rdi)
 0.00 :         86f62:       mov    %rdx,-0x16(%rdi)
 0.00 :         86f66:       mov    %rdx,-0xe(%rdi)
 0.00 :         86f6a:       mov    %edx,-0x6(%rdi)
 0.00 :         86f6d:       mov    %dx,-0x2(%rdi)
 0.00 :         86f71:       retq
 0.00 :         86f72:       data32 data32 data32 data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :         86f80:       mov    %rdx,-0x8f(%rdi)
 0.00 :         86f87:       mov    %rdx,-0x87(%rdi)
 0.00 :         86f8e:       mov    %rdx,-0x7f(%rdi)
 0.00 :         86f92:       mov    %rdx,-0x77(%rdi)
 0.00 :         86f96:       mov    %rdx,-0x6f(%rdi)
 0.00 :         86f9a:       mov    %rdx,-0x67(%rdi)
 0.00 :         86f9e:       mov    %rdx,-0x5f(%rdi)
 0.00 :         86fa2:       mov    %rdx,-0x57(%rdi)
 0.00 :         86fa6:       mov    %rdx,-0x4f(%rdi)
 0.00 :         86faa:       mov    %rdx,-0x47(%rdi)
 0.00 :         86fae:       mov    %rdx,-0x3f(%rdi)
 0.00 :         86fb2:       mov    %rdx,-0x37(%rdi)
 0.00 :         86fb6:       mov    %rdx,-0x2f(%rdi)
 0.00 :         86fba:       mov    %rdx,-0x27(%rdi)
 0.00 :         86fbe:       mov    %rdx,-0x1f(%rdi)
 0.00 :         86fc2:       mov    %rdx,-0x17(%rdi)
 0.00 :         86fc6:       mov    %rdx,-0xf(%rdi)
 0.00 :         86fca:       mov    %edx,-0x7(%rdi)
 0.00 :         86fcd:       mov    %dx,-0x3(%rdi)
 0.00 :         86fd1:       mov    %dl,-0x1(%rdi)
 0.00 :         86fd4:       retq
 0.00 :         86fd5:       data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :         86fe0:       mov    $0x10,%r10
 0.00 :         86fe7:       mov    %rdi,%r9
 0.00 :         86fea:       and    $0xf,%r9
 0.00 :         86fee:       sub    %r9,%r10
```

```
 0.00 :        86ff1:       and    $0xf,%r10
 0.00 :        86ff5:       add    %r10,%rdi
 0.00 :        86ff8:       sub    %r10,%r8
 0.00 :        86ffb:       lea    0x98(%rip),%r11        # 8709a <__GI_memset+0x3ca>
 0.00 :        87002:       lea    0xdc1b7(%rip),%rcx        # 1631c0 <result_type.7642+0x1a0>
 0.00 :        87009:       movswq (%rcx,%r10,2),%rcx
 0.00 :        8700e:       lea    (%rcx,%r11,1),%r11
 0.00 :        87012:       jmpq   *%r11
 0.00 :        87015:       data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :        87020:       mov    %dl,-0xd(%rdi)
 0.00 :        87023:       mov    %edx,-0xc(%rdi)
 0.00 :        87026:       mov    %rdx,-0x8(%rdi)
 0.00 :        8702a:       jmp    8709a <__GI_memset+0x3ca>
 0.00 :        8702c:       nopl   0x0(%rax)
 0.00 :        87030:       mov    %dl,-0x9(%rdi)
 0.00 :        87033:       mov    %rdx,-0x8(%rdi)
 0.00 :        87037:       jmp    8709a <__GI_memset+0x3ca>
 0.00 :        87039:       nopl   0x0(%rax)
 0.00 :        87040:       mov    %dl,-0x1(%rdi)
 0.00 :        87043:       jmp    8709a <__GI_memset+0x3ca>
 0.00 :        87045:       data32 nopw %cs:0x0(%rax,%rax,1)
 0.00 :        87050:       mov    %dl,-0xb(%rdi)
 0.00 :        87053:       mov    %dx,-0xa(%rdi)
 0.00 :        87057:       mov    %rdx,-0x8(%rdi)
 0.00 :        8705b:       jmp    8709a <__GI_memset+0x3ca>
 0.00 :        8705d:       nopl   (%rax)
 0.00 :        87060:       mov    %dl,-0x3(%rdi)
 0.00 :        87063:       mov    %dx,-0x2(%rdi)
 0.00 :        87067:       jmp    8709a <__GI_memset+0x3ca>
 0.00 :        87069:       nopl   0x0(%rax)
 0.00 :        87070:       mov    %dl,-0x5(%rdi)
 0.00 :        87073:       mov    %edx,-0x4(%rdi)
 0.00 :        87076:       jmp    8709a <__GI_memset+0x3ca>
 0.00 :        87078:       nopl   0x0(%rax,%rax,1)
 0.00 :        87080:       mov    %dl,-0xf(%rdi)
 0.00 :        87083:       mov    %dx,-0xe(%rdi)
 0.00 :        87087:       mov    %edx,-0xc(%rdi)
 0.00 :        8708a:       mov    %rdx,-0x8(%rdi)
 0.00 :        8708e:       jmp    8709a <__GI_memset+0x3ca>
 0.00 :        87090:       mov    %dl,-0x7(%rdi)
 0.00 :        87093:       mov    %dx,-0x6(%rdi)
 0.00 :        87097:       mov    %edx,-0x4(%rdi)
 0.00 :        8709a:       movq   %rdx,%xmm0
 0.00 :        8709f:       punpcklqdq %xmm0,%xmm0
 0.00 :        870a3:       cmp    $0xb0,%r8
 0.00 :        870aa:       jae    87580 <__GI_memset+0x8b0>
 0.00 :        870b0:       add    %r8,%rdi
 0.00 :        870b3:       lea    0x53(%rip),%r9        # 8710d <__GI_memset+0x43d>
 0.00 :        870ba:       lea    0xdc11f(%rip),%rcx        # 1631e0 <result_type.7642+0x1c0>
 0.00 :        870c1:       movswq (%rcx,%r8,2),%rcx
 0.00 :        870c6:       lea    (%rcx,%r9,1),%r9
 0.00 :        870ca:       jmpq   *%r9
 0.00 :        870cd:       movdqa %xmm0,-0xb0(%rdi)
 0.00 :        870d5:       movdqa %xmm0,-0xa0(%rdi)
 0.00 :        870dd:       movdqa %xmm0,-0x90(%rdi)
 0.00 :        870e5:       movdqa %xmm0,-0x80(%rdi)
 0.00 :        870ea:       movdqa %xmm0,-0x70(%rdi)
 0.00 :        870ef:       movdqa %xmm0,-0x60(%rdi)
 0.00 :        870f4:       movdqa %xmm0,-0x50(%rdi)
 0.00 :        870f9:       movdqa %xmm0,-0x40(%rdi)
 0.00 :        870fe:       movdqa %xmm0,-0x30(%rdi)
 0.00 :        87103:       movdqa %xmm0,-0x20(%rdi)
 0.00 :        87108:       movdqa %xmm0,-0x10(%rdi)
 0.00 :        8710d:       retq
```

```
 0.00 :        8710e:       movdqa %xmm0,-0xb1(%rdi)
 0.00 :        87116:       movdqa %xmm0,-0xa1(%rdi)
 0.00 :        8711e:       movdqa %xmm0,-0x91(%rdi)
 0.00 :        87126:       movdqa %xmm0,-0x81(%rdi)
 0.00 :        8712e:       movdqa %xmm0,-0x71(%rdi)
 0.00 :        87133:       movdqa %xmm0,-0x61(%rdi)
 0.00 :        87138:       movdqa %xmm0,-0x51(%rdi)
 0.00 :        8713d:       movdqa %xmm0,-0x41(%rdi)
 0.00 :        87142:       movdqa %xmm0,-0x31(%rdi)
 0.00 :        87147:       movdqa %xmm0,-0x21(%rdi)
 0.00 :        8714c:       movdqa %xmm0,-0x11(%rdi)
 0.00 :        87151:       mov    %dl,-0x1(%rdi)
 0.00 :        87154:       retq
 0.00 :        87155:       movdqa %xmm0,-0xb2(%rdi)
 0.00 :        8715d:       movdqa %xmm0,-0xa2(%rdi)
 0.00 :        87165:       movdqa %xmm0,-0x92(%rdi)
 0.00 :        8716d:       movdqa %xmm0,-0x82(%rdi)
 0.00 :        87175:       movdqa %xmm0,-0x72(%rdi)
 0.00 :        8717a:       movdqa %xmm0,-0x62(%rdi)
 0.00 :        8717f:       movdqa %xmm0,-0x52(%rdi)
 0.00 :        87184:       movdqa %xmm0,-0x42(%rdi)
 0.00 :        87189:       movdqa %xmm0,-0x32(%rdi)
 0.00 :        8718e:       movdqa %xmm0,-0x22(%rdi)
 0.00 :        87193:       movdqa %xmm0,-0x12(%rdi)
 0.00 :        87198:       mov    %dx,-0x2(%rdi)
 0.00 :        8719c:       retq
 0.00 :        8719d:       movdqa %xmm0,-0xb3(%rdi)
 0.00 :        871a5:       movdqa %xmm0,-0xa3(%rdi)
 0.00 :        871ad:       movdqa %xmm0,-0x93(%rdi)
 0.00 :        871b5:       movdqa %xmm0,-0x83(%rdi)
 0.00 :        871bd:       movdqa %xmm0,-0x73(%rdi)
 0.00 :        871c2:       movdqa %xmm0,-0x63(%rdi)
 0.00 :        871c7:       movdqa %xmm0,-0x53(%rdi)
 0.00 :        871cc:       movdqa %xmm0,-0x43(%rdi)
 0.00 :        871d1:       movdqa %xmm0,-0x33(%rdi)
 0.00 :        871d6:       movdqa %xmm0,-0x23(%rdi)
 0.00 :        871db:       movdqa %xmm0,-0x13(%rdi)
 0.00 :        871e0:       mov    %dx,-0x3(%rdi)
 0.00 :        871e4:       mov    %dl,-0x1(%rdi)
 0.00 :        871e7:       retq
 0.00 :        871e8:       movdqa %xmm0,-0xb4(%rdi)
 0.00 :        871f0:       movdqa %xmm0,-0xa4(%rdi)
 0.00 :        871f8:       movdqa %xmm0,-0x94(%rdi)
 0.00 :        87200:       movdqa %xmm0,-0x84(%rdi)
 0.00 :        87208:       movdqa %xmm0,-0x74(%rdi)
 0.00 :        8720d:       movdqa %xmm0,-0x64(%rdi)
 0.00 :        87212:       movdqa %xmm0,-0x54(%rdi)
 0.00 :        87217:       movdqa %xmm0,-0x44(%rdi)
 0.00 :        8721c:       movdqa %xmm0,-0x34(%rdi)
 0.00 :        87221:       movdqa %xmm0,-0x24(%rdi)
 0.00 :        87226:       movdqa %xmm0,-0x14(%rdi)
 0.00 :        8722b:       mov    %edx,-0x4(%rdi)
 0.00 :        8722e:       retq
 0.00 :        8722f:       movdqa %xmm0,-0xb5(%rdi)
 0.00 :        87237:       movdqa %xmm0,-0xa5(%rdi)
 0.00 :        8723f:       movdqa %xmm0,-0x95(%rdi)
 0.00 :        87247:       movdqa %xmm0,-0x85(%rdi)
 0.00 :        8724f:       movdqa %xmm0,-0x75(%rdi)
 0.00 :        87254:       movdqa %xmm0,-0x65(%rdi)
 0.00 :        87259:       movdqa %xmm0,-0x55(%rdi)
 0.00 :        8725e:       movdqa %xmm0,-0x45(%rdi)
 0.00 :        87263:       movdqa %xmm0,-0x35(%rdi)
 0.00 :        87268:       movdqa %xmm0,-0x25(%rdi)
 0.00 :        8726d:       movdqa %xmm0,-0x15(%rdi)
```

```
 0.00 :        87272:       mov    %edx,-0x5(%rdi)
 0.00 :        87275:       mov    %dl,-0x1(%rdi)
 0.00 :        87278:       retq
 0.00 :        87279:       movdqa %xmm0,-0xb6(%rdi)
 0.00 :        87281:       movdqa %xmm0,-0xa6(%rdi)
 0.00 :        87289:       movdqa %xmm0,-0x96(%rdi)
 0.00 :        87291:       movdqa %xmm0,-0x86(%rdi)
 0.00 :        87299:       movdqa %xmm0,-0x76(%rdi)
 0.00 :        8729e:       movdqa %xmm0,-0x66(%rdi)
 0.00 :        872a3:       movdqa %xmm0,-0x56(%rdi)
 0.00 :        872a8:       movdqa %xmm0,-0x46(%rdi)
 0.00 :        872ad:       movdqa %xmm0,-0x36(%rdi)
 0.00 :        872b2:       movdqa %xmm0,-0x26(%rdi)
 0.00 :        872b7:       movdqa %xmm0,-0x16(%rdi)
 0.00 :        872bc:       mov    %edx,-0x6(%rdi)
 0.00 :        872bf:       mov    %dx,-0x2(%rdi)
 0.00 :        872c3:       retq
 0.00 :        872c4:       movdqa %xmm0,-0xb7(%rdi)
 0.00 :        872cc:       movdqa %xmm0,-0xa7(%rdi)
 0.00 :        872d4:       movdqa %xmm0,-0x97(%rdi)
 0.00 :        872dc:       movdqa %xmm0,-0x87(%rdi)
 0.00 :        872e4:       movdqa %xmm0,-0x77(%rdi)
 0.00 :        872e9:       movdqa %xmm0,-0x67(%rdi)
 0.00 :        872ee:       movdqa %xmm0,-0x57(%rdi)
 0.00 :        872f3:       movdqa %xmm0,-0x47(%rdi)
 0.00 :        872f8:       movdqa %xmm0,-0x37(%rdi)
 0.00 :        872fd:       movdqa %xmm0,-0x27(%rdi)
 0.00 :        87302:       movdqa %xmm0,-0x17(%rdi)
 0.00 :        87307:       mov    %edx,-0x7(%rdi)
 0.00 :        8730a:       mov    %dx,-0x3(%rdi)
 0.00 :        8730e:       mov    %dl,-0x1(%rdi)
 0.00 :        87311:       retq
 0.00 :        87312:       movdqa %xmm0,-0xb8(%rdi)
 0.00 :        8731a:       movdqa %xmm0,-0xa8(%rdi)
 0.00 :        87322:       movdqa %xmm0,-0x98(%rdi)
 0.00 :        8732a:       movdqa %xmm0,-0x88(%rdi)
 0.00 :        87332:       movdqa %xmm0,-0x78(%rdi)
 0.00 :        87337:       movdqa %xmm0,-0x68(%rdi)
 0.00 :        8733c:       movdqa %xmm0,-0x58(%rdi)
 0.00 :        87341:       movdqa %xmm0,-0x48(%rdi)
 0.00 :        87346:       movdqa %xmm0,-0x38(%rdi)
 0.00 :        8734b:       movdqa %xmm0,-0x28(%rdi)
 0.00 :        87350:       movdqa %xmm0,-0x18(%rdi)
 0.00 :        87355:       mov    %rdx,-0x8(%rdi)
 0.00 :        87359:       retq
 0.00 :        8735a:       movdqa %xmm0,-0xb9(%rdi)
 0.00 :        87362:       movdqa %xmm0,-0xa9(%rdi)
 0.00 :        8736a:       movdqa %xmm0,-0x99(%rdi)
 0.00 :        87372:       movdqa %xmm0,-0x89(%rdi)
 0.00 :        8737a:       movdqa %xmm0,-0x79(%rdi)
 0.00 :        8737f:       movdqa %xmm0,-0x69(%rdi)
 0.00 :        87384:       movdqa %xmm0,-0x59(%rdi)
 0.00 :        87389:       movdqa %xmm0,-0x49(%rdi)
 0.00 :        8738e:       movdqa %xmm0,-0x39(%rdi)
 0.00 :        87393:       movdqa %xmm0,-0x29(%rdi)
 0.00 :        87398:       movdqa %xmm0,-0x19(%rdi)
 0.00 :        8739d:       mov    %rdx,-0x9(%rdi)
 0.00 :        873a1:       mov    %dl,-0x1(%rdi)
 0.00 :        873a4:       retq
 0.00 :        873a5:       movdqa %xmm0,-0xba(%rdi)
 0.00 :        873ad:       movdqa %xmm0,-0xaa(%rdi)
 0.00 :        873b5:       movdqa %xmm0,-0x9a(%rdi)
 0.00 :        873bd:       movdqa %xmm0,-0x8a(%rdi)
 0.00 :        873c5:       movdqa %xmm0,-0x7a(%rdi)
```

```
 0.00 :        873ca:       movdqa %xmm0,-0x6a(%rdi)
 0.00 :        873cf:       movdqa %xmm0,-0x5a(%rdi)
 0.00 :        873d4:       movdqa %xmm0,-0x4a(%rdi)
 0.00 :        873d9:       movdqa %xmm0,-0x3a(%rdi)
 0.00 :        873de:       movdqa %xmm0,-0x2a(%rdi)
 0.00 :        873e3:       movdqa %xmm0,-0x1a(%rdi)
 0.00 :        873e8:       mov    %rdx,-0xa(%rdi)
 0.00 :        873ec:       mov    %dx,-0x2(%rdi)
 0.00 :        873f0:       retq
 0.00 :        873f1:       movdqa %xmm0,-0xbb(%rdi)
 0.00 :        873f9:       movdqa %xmm0,-0xab(%rdi)
 0.00 :        87401:       movdqa %xmm0,-0x9b(%rdi)
 0.00 :        87409:       movdqa %xmm0,-0x8b(%rdi)
 0.00 :        87411:       movdqa %xmm0,-0x7b(%rdi)
 0.00 :        87416:       movdqa %xmm0,-0x6b(%rdi)
 0.00 :        8741b:       movdqa %xmm0,-0x5b(%rdi)
 0.00 :        87420:       movdqa %xmm0,-0x4b(%rdi)
 0.00 :        87425:       movdqa %xmm0,-0x3b(%rdi)
 0.00 :        8742a:       movdqa %xmm0,-0x2b(%rdi)
 0.00 :        8742f:       movdqa %xmm0,-0x1b(%rdi)
 0.00 :        87434:       mov    %rdx,-0xb(%rdi)
 0.00 :        87438:       mov    %dx,-0x3(%rdi)
 0.00 :        8743c:       mov    %dl,-0x1(%rdi)
 0.00 :        8743f:       retq
 0.00 :        87440:       movdqa %xmm0,-0xbc(%rdi)
 0.00 :        87448:       movdqa %xmm0,-0xac(%rdi)
 0.00 :        87450:       movdqa %xmm0,-0x9c(%rdi)
 0.00 :        87458:       movdqa %xmm0,-0x8c(%rdi)
 0.00 :        87460:       movdqa %xmm0,-0x7c(%rdi)
 0.00 :        87465:       movdqa %xmm0,-0x6c(%rdi)
 0.00 :        8746a:       movdqa %xmm0,-0x5c(%rdi)
 0.00 :        8746f:       movdqa %xmm0,-0x4c(%rdi)
 0.00 :        87474:       movdqa %xmm0,-0x3c(%rdi)
 0.00 :        87479:       movdqa %xmm0,-0x2c(%rdi)
 0.00 :        8747e:       movdqa %xmm0,-0x1c(%rdi)
 0.00 :        87483:       mov    %rdx,-0xc(%rdi)
 0.00 :        87487:       mov    %edx,-0x4(%rdi)
 0.00 :        8748a:       retq
 0.00 :        8748b:       movdqa %xmm0,-0xbd(%rdi)
 0.00 :        87493:       movdqa %xmm0,-0xad(%rdi)
 0.00 :        8749b:       movdqa %xmm0,-0x9d(%rdi)
 0.00 :        874a3:       movdqa %xmm0,-0x8d(%rdi)
 0.00 :        874ab:       movdqa %xmm0,-0x7d(%rdi)
 0.00 :        874b0:       movdqa %xmm0,-0x6d(%rdi)
 0.00 :        874b5:       movdqa %xmm0,-0x5d(%rdi)
 0.00 :        874ba:       movdqa %xmm0,-0x4d(%rdi)
 0.00 :        874bf:       movdqa %xmm0,-0x3d(%rdi)
 0.00 :        874c4:       movdqa %xmm0,-0x2d(%rdi)
 0.00 :        874c9:       movdqa %xmm0,-0x1d(%rdi)
 0.00 :        874ce:       mov    %rdx,-0xd(%rdi)
 0.00 :        874d2:       mov    %edx,-0x5(%rdi)
 0.00 :        874d5:       mov    %dl,-0x1(%rdi)
 0.00 :        874d8:       retq
 0.00 :        874d9:       movdqa %xmm0,-0xbe(%rdi)
 0.00 :        874e1:       movdqa %xmm0,-0xae(%rdi)
 0.00 :        874e9:       movdqa %xmm0,-0x9e(%rdi)
 0.00 :        874f1:       movdqa %xmm0,-0x8e(%rdi)
 0.00 :        874f9:       movdqa %xmm0,-0x7e(%rdi)
 0.00 :        874fe:       movdqa %xmm0,-0x6e(%rdi)
 0.00 :        87503:       movdqa %xmm0,-0x5e(%rdi)
 0.00 :        87508:       movdqa %xmm0,-0x4e(%rdi)
 0.00 :        8750d:       movdqa %xmm0,-0x3e(%rdi)
 0.00 :        87512:       movdqa %xmm0,-0x2e(%rdi)
 0.00 :        87517:       movdqa %xmm0,-0x1e(%rdi)
```

```
  0.00 :         8751c:       mov    %rdx,-0xe(%rdi)
  0.00 :         87520:       mov    %edx,-0x6(%rdi)
  0.00 :         87523:       mov    %dx,-0x2(%rdi)
  0.00 :         87527:       retq
  0.00 :         87528:       movdqa %xmm0,-0xbf(%rdi)
  0.00 :         87530:       movdqa %xmm0,-0xaf(%rdi)
  0.00 :         87538:       movdqa %xmm0,-0x9f(%rdi)
  0.00 :         87540:       movdqa %xmm0,-0x8f(%rdi)
  0.00 :         87548:       movdqa %xmm0,-0x7f(%rdi)
  0.00 :         8754d:       movdqa %xmm0,-0x6f(%rdi)
  0.00 :         87552:       movdqa %xmm0,-0x5f(%rdi)
  0.00 :         87557:       movdqa %xmm0,-0x4f(%rdi)
  0.00 :         8755c:       movdqa %xmm0,-0x3f(%rdi)
  0.00 :         87561:       movdqa %xmm0,-0x2f(%rdi)
  0.00 :         87566:       movdqa %xmm0,-0x1f(%rdi)
  0.00 :         8756b:       mov    %rdx,-0xf(%rdi)
  0.00 :         8756f:       mov    %edx,-0x7(%rdi)
  0.00 :         87572:       mov    %dx,-0x3(%rdi)
  0.00 :         87576:       mov    %dl,-0x1(%rdi)
  0.00 :         87579:       retq
  0.00 :         8757a:       nopw   0x0(%rax,%rax,1)
  0.00 :         87580:       mov    0x31fc59(%rip),%r9d        # 3a71e0 <__x86_64_shared_cache_size>
  0.00 :         87587:       cmp    %r9,%r8
  0.00 :         8758a:       ja     875f0 <__GI_memset+0x920>
  0.00 :         8758c:       nopl   0x0(%rax)
  0.00 :         87590:       lea    -0x80(%r8),%r8
  0.00 :         87594:       cmp    $0x80,%r8
  0.00 :         8759b:       movdqa %xmm0,(%rdi)
  0.00 :         8759f:       movdqa %xmm0,0x10(%rdi)
  0.00 :         875a4:       movdqa %xmm0,0x20(%rdi)
  0.00 :         875a9:       movdqa %xmm0,0x30(%rdi)
  0.00 :         875ae:       movdqa %xmm0,0x40(%rdi)
  0.00 :         875b3:       movdqa %xmm0,0x50(%rdi)
  0.00 :         875b8:       movdqa %xmm0,0x60(%rdi)
100.00 :         875bd:       movdqa %xmm0,0x70(%rdi)
  0.00 :         875c2:       lea    0x80(%rdi),%rdi
  0.00 :         875c9:       jae    87590 <__GI_memset+0x8c0>
  0.00 :         875cb:       add    %r8,%rdi
  0.00 :         875ce:       lea    -0x4c8(%rip),%r11        # 8710d <__GI_memset+0x43d>
  0.00 :         875d5:       lea    0xdbc04(%rip),%rcx        # 1631e0 <result_type.7642+0x1c0>
  0.00 :         875dc:       movswq (%rcx,%r8,2),%rcx
  0.00 :         875e1:       lea    (%rcx,%r11,1),%r11
  0.00 :         875e5:       jmpq   *%r11
  0.00 :         875e8:       nopl   0x0(%rax,%rax,1)
  0.00 :         875f0:       cmp    $0x0,%r9
  0.00 :         875f4:       je     87590 <__GI_memset+0x8c0>
  0.00 :         875f6:       jmp    87600 <__GI_memset+0x930>
  0.00 :         875f8:       nopl   0x0(%rax,%rax,1)
  0.00 :         87600:       lea    -0x80(%r8),%r8
  0.00 :         87604:       cmp    $0x80,%r8
  0.00 :         8760b:       movntdq %xmm0,(%rdi)
  0.00 :         8760f:       movntdq %xmm0,0x10(%rdi)
  0.00 :         87614:       movntdq %xmm0,0x20(%rdi)
  0.00 :         87619:       movntdq %xmm0,0x30(%rdi)
  0.00 :         8761e:       movntdq %xmm0,0x40(%rdi)
  0.00 :         87623:       movntdq %xmm0,0x50(%rdi)
  0.00 :         87628:       movntdq %xmm0,0x60(%rdi)
  0.00 :         8762d:       movntdq %xmm0,0x70(%rdi)
  0.00 :         87632:       lea    0x80(%rdi),%rdi
  0.00 :         87639:       jae    87600 <__GI_memset+0x930>
  0.00 :         8763b:       sfence
  0.00 :         8763e:       add    %r8,%rdi
  0.00 :         87641:       lea    -0x53b(%rip),%r11        # 8710d <__GI_memset+0x43d>
  0.00 :         87648:       lea    0xdbb91(%rip),%rcx        # 1631e0 <result_type.7642+0x1c0>
```

```
 0.00 :         8764f:       movswq (%rcx,%r8,2),%rcx
 0.00 :         87654:       lea    (%rcx,%r11,1),%r11
 0.00 :         87658:       jmpq   *%r11
 0.00 :         8765b:       nopl   0x0(%rax,%rax,1)
 0.00 :         87660:       cmp    $0x2000,%r8
 0.00 :         87667:       jae    876f0 <__GI_memset+0xa20>
 0.00 :         8766d:       nopl   (%rax)
 0.00 :         87670:       mov    %r8,%rcx
 0.00 :         87673:       shr    $0x7,%rcx
 0.00 :         87677:       je     876cb <__GI_memset+0x9fb>
 0.00 :         87679:       nopl   0x0(%rax)
 0.00 :         87680:       dec    %rcx
 0.00 :         87683:       mov    %rdx,(%rdi)
 0.00 :         87686:       mov    %rdx,0x8(%rdi)
 0.00 :         8768a:       mov    %rdx,0x10(%rdi)
 0.00 :         8768e:       mov    %rdx,0x18(%rdi)
 0.00 :         87692:       mov    %rdx,0x20(%rdi)
 0.00 :         87696:       mov    %rdx,0x28(%rdi)
 0.00 :         8769a:       mov    %rdx,0x30(%rdi)
 0.00 :         8769e:       mov    %rdx,0x38(%rdi)
 0.00 :         876a2:       mov    %rdx,0x40(%rdi)
 0.00 :         876a6:       mov    %rdx,0x48(%rdi)
 0.00 :         876aa:       mov    %rdx,0x50(%rdi)
 0.00 :         876ae:       mov    %rdx,0x58(%rdi)
 0.00 :         876b2:       mov    %rdx,0x60(%rdi)
 0.00 :         876b6:       mov    %rdx,0x68(%rdi)
 0.00 :         876ba:       mov    %rdx,0x70(%rdi)
 0.00 :         876be:       mov    %rdx,0x78(%rdi)
 0.00 :         876c2:       lea    0x80(%rdi),%rdi
 0.00 :         876c9:       jne    87680 <__GI_memset+0x9b0>
 0.00 :         876cb:       and    $0x7f,%r8d
 0.00 :         876cf:       lea    (%rdi,%r8,1),%rdi
 0.00 :         876d3:       lea    -0x9be(%rip),%r11        # 86d1c <__GI_memset+0x4c>
 0.00 :         876da:       lea    0xdb9af(%rip),%rcx        # 163090 <result_type.7642+0x70>
 0.00 :         876e1:       movswq (%rcx,%r8,2),%rcx
 0.00 :         876e6:       lea    (%rcx,%r11,1),%r11
 0.00 :         876ea:       jmpq   *%r11
 0.00 :         876ed:       nopl   (%rax)
 0.00 :         876f0:       mov    0x31fae9(%rip),%r9d        # 3a71e0 <__x86_64_shared_cache_size>
 0.00 :         876f7:       cmp    %r8,%r9
 0.00 :         876fa:       cmova  %r8,%r9
 0.00 :         876fe:       jbe    87710 <__GI_memset+0xa40>
 0.00 :         87700:       cmp    $0x10000,%r8
 0.00 :         87707:       jae    87670 <__GI_memset+0x9a0>
 0.00 :         8770d:       nopl   (%rax)
 0.00 :         87710:       mov    %r9,%rcx
 0.00 :         87713:       and    $0xfffffffffffffff8,%r9
 0.00 :         87717:       shr    $0x3,%rcx
 0.00 :         8771b:       je     87724 <__GI_memset+0xa54>
 0.00 :         8771d:       xchg   %rax,%rdx
 0.00 :         8771f:       rep stos %rax,%es:(%rdi)
 0.00 :         87722:       xchg   %rax,%rdx
 0.00 :         87724:       sub    %r9,%r8
 0.00 :         87727:       ja     87750 <__GI_memset+0xa80>
 0.00 :         87729:       and    $0x7,%r8d
 0.00 :         8772d:       lea    (%rdi,%r8,1),%rdi
 0.00 :         87731:       lea    -0xa1c(%rip),%r11        # 86d1c <__GI_memset+0x4c>
 0.00 :         87738:       lea    0xdb951(%rip),%rcx        # 163090 <result_type.7642+0x70>
 0.00 :         8773f:       movswq (%rcx,%r8,2),%rcx
 0.00 :         87744:       lea    (%rcx,%r11,1),%r11
 0.00 :         87748:       jmpq   *%r11
 0.00 :         8774b:       nopl   0x0(%rax,%rax,1)
 0.00 :         87750:       mov    %r8,%rcx
 0.00 :         87753:       shr    $0x7,%rcx
```

```
    0.00 :          87757:       je     877be <__GI_memset+0xaee>
    0.00 :          87759:       nopl   0x0(%rax)
    0.00 :          87760:       dec    %rcx
    0.00 :          87763:       movnti %rdx,(%rdi)
    0.00 :          87767:       movnti %rdx,0x8(%rdi)
    0.00 :          8776c:       movnti %rdx,0x10(%rdi)
    0.00 :          87771:       movnti %rdx,0x18(%rdi)
    0.00 :          87776:       movnti %rdx,0x20(%rdi)
    0.00 :          8777b:       movnti %rdx,0x28(%rdi)
    0.00 :          87780:       movnti %rdx,0x30(%rdi)
    0.00 :          87785:       movnti %rdx,0x38(%rdi)
    0.00 :          8778a:       movnti %rdx,0x40(%rdi)
    0.00 :          8778f:       movnti %rdx,0x48(%rdi)
    0.00 :          87794:       movnti %rdx,0x50(%rdi)
    0.00 :          87799:       movnti %rdx,0x58(%rdi)
    0.00 :          8779e:       movnti %rdx,0x60(%rdi)
    0.00 :          877a3:       movnti %rdx,0x68(%rdi)
    0.00 :          877a8:       movnti %rdx,0x70(%rdi)
    0.00 :          877ad:       movnti %rdx,0x78(%rdi)
    0.00 :          877b2:       lea    0x80(%rdi),%rdi
    0.00 :          877b9:       jne    87760 <__GI_memset+0xa90>
    0.00 :          877bb:       sfence
    0.00 :          877be:       and    $0x7f,%r8d
    0.00 :          877c2:       lea    (%rdi,%r8,1),%rdi
    0.00 :          877c6:       lea    -0xab1(%rip),%r11        # 86d1c <__GI_memset+0x4c>
    0.00 :          877cd:       lea    0xdb8bc(%rip),%rcx        # 163090 <result_type.7642+0x70>
    0.00 :          877d4:       movswq (%rcx,%r8,2),%rcx
    0.00 :          877d9:       lea    (%rcx,%r11,1),%r11
    0.00 :          877dd:       jmpq   *%r11
 Percent |       Source code & Disassembly of libQtCore.so.4.8.4
-----------------------------------------------
         :
         :
         :
         :       Disassembly of section .text:
         :
         :       000000000007d8e0 <QByteArray::realloc(int)>:
  100.00 :          7d8e0:       mov    %rbp,-0x18(%rsp)
    0.00 :          7d8e5:       mov    %r12,-0x10(%rsp)
    0.00 :          7d8ea:       mov    %rdi,%rbp
    0.00 :          7d8ed:       mov    %rbx,-0x20(%rsp)
    0.00 :          7d8f2:       mov    %r13,-0x8(%rsp)
    0.00 :          7d8f7:       sub    $0x28,%rsp
    0.00 :          7d8fb:       mov    (%rdi),%rdi
    0.00 :          7d8fe:       movslq %esi,%r12
    0.00 :          7d901:       mov    (%rdi),%eax
    0.00 :          7d903:       cmp    $0x1,%eax
    0.00 :          7d906:       je     7d990 <QByteArray::realloc(int)+0xb0>
    0.00 :          7d90c:       movslq %r12d,%rdi
    0.00 :          7d90f:       add    $0x20,%rdi
    0.00 :          7d913:       callq  74c10 <qMalloc(unsigned long)>
    0.00 :          7d918:       test   %rax,%rax
    0.00 :          7d91b:       mov    %rax,%rbx
    0.00 :          7d91e:       je     7d9d8 <QByteArray::realloc(int)+0xf8>
    0.00 :          7d924:       mov    0x0(%rbp),%rax
    0.00 :          7d928:       lea    0x18(%rbx),%r13
    0.00 :          7d92c:       mov    %r13,%rdi
    0.00 :          7d92f:       movslq 0x8(%rax),%rdx
    0.00 :          7d933:       mov    0x10(%rax),%rsi
    0.00 :          7d937:       cmp    %edx,%r12d
    0.00 :          7d93a:       cmovle %r12,%rdx
    0.00 :          7d93e:       mov    %edx,0x8(%rbx)
    0.00 :          7d941:       callq  5d560 <memcpy@plt>
    0.00 :          7d946:       movslq 0x8(%rbx),%rax
```

```
    0.00 :          7d94a:       mov    %r12d,0x4(%rbx)
    0.00 :          7d94e:       mov    %r13,0x10(%rbx)
    0.00 :          7d952:       movb   $0x0,0x18(%rbx,%rax,1)
    0.00 :          7d957:       movl   $0x1,(%rbx)
    0.00 :          7d95d:       mov    0x0(%rbp),%rax
    0.00 :          7d961:       lock decl (%rax)
    0.00 :          7d964:       setne  %dl
    0.00 :          7d967:       test   %dl,%dl
    0.00 :          7d969:       je     7d9c8 <QByteArray::realloc(int)+0xe8>
    0.00 :          7d96b:       mov    %rbx,0x0(%rbp)
    0.00 :          7d96f:       mov    0x18(%rsp),%r12
    0.00 :          7d974:       mov    0x8(%rsp),%rbx
    0.00 :          7d979:       mov    0x10(%rsp),%rbp
    0.00 :          7d97e:       mov    0x20(%rsp),%r13
    0.00 :          7d983:       add    $0x28,%rsp
    0.00 :          7d987:       retq
    0.00 :          7d988:       nopl   0x0(%rax,%rax,1)
    0.00 :          7d990:       lea    0x18(%rdi),%rax
    0.00 :          7d994:       cmp    %rax,0x10(%rdi)
    0.00 :          7d998:       jne    7d90c <QByteArray::realloc(int)+0x2c>
    0.00 :          7d99e:       movslq %r12d,%rsi
    0.00 :          7d9a1:       add    $0x20,%rsi
    0.00 :          7d9a5:       callq  74c30 <qRealloc(void*, unsigned long)>
    0.00 :          7d9aa:       test   %rax,%rax
    0.00 :          7d9ad:       mov    %rax,%rbx
    0.00 :          7d9b0:       jne    7d9b7 <QByteArray::realloc(int)+0xd7>
    0.00 :          7d9b2:       callq  71070 <qBadAlloc()>
    0.00 :          7d9b7:       lea    0x18(%rbx),%rax
    0.00 :          7d9bb:       mov    %r12d,0x4(%rbx)
    0.00 :          7d9bf:       mov    %rax,0x10(%rbx)
    0.00 :          7d9c3:       jmp    7d96b <QByteArray::realloc(int)+0x8b>
    0.00 :          7d9c5:       nopl   (%rax)
    0.00 :          7d9c8:       mov    0x0(%rbp),%rdi
    0.00 :          7d9cc:       callq  74c20 <qFree(void*)>
    0.00 :          7d9d1:       jmp    7d96b <QByteArray::realloc(int)+0x8b>
    0.00 :          7d9d3:       nopl   0x0(%rax,%rax,1)
    0.00 :          7d9d8:       callq  71070 <qBadAlloc()>
    0.00 :          7d9dd:       nopl   (%rax)
    0.00 :          7d9e0:       jmpq   7d924 <QByteArray::realloc(int)+0x44>
 Percent |       Source code & Disassembly of libQtCore.so.4.8.4
-----------------------------------------------
         :
         :
         :
         :       Disassembly of section .text:
         :
         :       0000000000099760 <QListData::remove(int, int)>:
    0.00 :          99760:       push   %rbp
  100.00 :          99761:       mov    %edx,%ecx
    0.00 :          99763:       mov    %rdi,%rbp
    0.00 :          99766:       shr    $0x1f,%ecx
    0.00 :          99769:       push   %rbx
    0.00 :          9976a:       add    %edx,%ecx
    0.00 :          9976c:       mov    %edx,%ebx
    0.00 :          9976e:       sar    %ecx
    0.00 :          99770:       sub    $0x8,%rsp
    0.00 :          99774:       mov    (%rdi),%rax
    0.00 :          99777:       movslq 0x8(%rax),%r8
    0.00 :          9977b:       mov    0xc(%rax),%edi
    0.00 :          9977e:       mov    %edi,%edx
    0.00 :          99780:       lea    (%r8,%rsi,1),%r9d
    0.00 :          99784:       add    %r9d,%ecx
    0.00 :          99787:       mov    %ecx,%r10d
    0.00 :          9978a:       sub    %ecx,%edx
```

```
    0.00 :          9978c:       sub    %r8d,%r10d
    0.00 :          9978f:       cmp    %edx,%r10d
    0.00 :          99792:       jl     997d0 <QListData::remove(int, int)+0x70>
    0.00 :          99794:       movslq %r9d,%rcx
    0.00 :          99797:       sub    %r9d,%edi
    0.00 :          9979a:       movslq %ebx,%rsi
    0.00 :          9979d:       sub    %ebx,%edi
    0.00 :          9979f:       lea    0x2(%rcx,%rsi,1),%rsi
    0.00 :          997a4:       movslq %edi,%rdi
    0.00 :          997a7:       lea    0x0(,%rdi,8),%rdx
    0.00 :          997af:       lea    0x8(%rax,%rsi,8),%rsi
    0.00 :          997b4:       lea    0x18(%rax,%rcx,8),%rdi
    0.00 :          997b9:       callq  5dbc0 <memmove@plt>
    0.00 :          997be:       mov    0x0(%rbp),%rax
    0.00 :          997c2:       sub    %ebx,0xc(%rax)
    0.00 :          997c5:       add    $0x8,%rsp
    0.00 :          997c9:       pop    %rbx
    0.00 :          997ca:       pop    %rbp
    0.00 :          997cb:       retq
    0.00 :          997cc:       nopl   0x0(%rax)
    0.00 :          997d0:       movslq %ebx,%rcx
    0.00 :          997d3:       movslq %esi,%rsi
    0.00 :          997d6:       lea    0x2(%r8,%rcx,1),%rcx
    0.00 :          997db:       lea    0x0(,%rsi,8),%rdx
    0.00 :          997e3:       lea    0x18(%rax,%r8,8),%rsi
    0.00 :          997e8:       lea    0x8(%rax,%rcx,8),%rdi
    0.00 :          997ed:       callq  5dbc0 <memmove@plt>
    0.00 :          997f2:       mov    0x0(%rbp),%rax
    0.00 :          997f6:       add    %ebx,0x8(%rax)
    0.00 :          997f9:       add    $0x8,%rsp
    0.00 :          997fd:       pop    %rbx
    0.00 :          997fe:       pop    %rbp
    0.00 :          997ff:       retq
 Percent |       Source code & Disassembly of ld-2.17.so
-----------------------------------------------
         :
         :
         :
         :       Disassembly of section .text:
         :
         :       00000000000096d0 <_dl_lookup_symbol_x>:
    0.00 :          96d0:       push   %rbp
    0.00 :          96d1:       mov    %rsp,%rbp
    0.00 :          96d4:       push   %r15
    0.00 :          96d6:       mov    %rsi,%r15
    0.00 :          96d9:       push   %r14
    0.00 :          96db:       push   %r13
    0.00 :          96dd:       mov    %r9d,%r13d
    0.00 :          96e0:       push   %r12
    0.00 :          96e2:       mov    %r8,%r12
    0.00 :          96e5:       push   %rbx
    0.00 :          96e6:       mov    %rdx,%rbx
    0.00 :          96e9:       sub    $0x118,%rsp
    0.00 :          96f0:       movzbl (%rdi),%eax
    0.00 :          96f3:       mov    %rdi,-0xb8(%rbp)
    0.00 :          96fa:       mov    %rcx,-0xd8(%rbp)
    0.00 :          9701:       test   %al,%al
    0.00 :          9703:       je     9968 <_dl_lookup_symbol_x+0x298>
    0.00 :          9709:       mov    %rdi,%rdx
    0.00 :          970c:       mov    $0x1505,%r14d
    0.00 :          9712:       nopw   0x0(%rax,%rax,1)
    0.00 :          9718:       mov    %r14,%rcx
    0.00 :          971b:       add    $0x1,%rdx
    0.00 :          971f:       shl    $0x5,%rcx
```

```
  0.00 :        9723:       add    %rcx,%r14
  0.00 :        9726:       add    %rax,%r14
100.00 :        9729:       movzbl (%rdx),%eax
  0.00 :        972c:       test   %al,%al
  0.00 :        972e:       jne    9718 <_dl_lookup_symbol_x+0x48>
  0.00 :        9730:       mov    %r14d,%r14d
  0.00 :        9733:       mov    %r14,-0xd0(%rbp)
  0.00 :        973a:       mov    $0xffffffff,%r11d
  0.00 :        9740:       addq   $0x1,0x219230(%rip)        # 222978 <_rtld_global+0x978>
  0.00 :        9748:       test   %r12,%r12
  0.00 :        974b:       mov    %r11,-0xb0(%rbp)
  0.00 :        9752:       movq   $0x0,-0xa0(%rbp)
  0.00 :        975d:       movq   $0x0,-0x98(%rbp)
  0.00 :        9768:       je     9777 <_dl_lookup_symbol_x+0xa7>
  0.00 :        976a:       testl  $0xfffffffa,0x10(%rbp)
  0.00 :        9771:       jne    9fa0 <_dl_lookup_symbol_x+0x8d0>
  0.00 :        9777:       cmpq   $0x0,0x18(%rbp)
  0.00 :        977c:       mov    -0xd8(%rbp),%rcx
  0.00 :        9783:       mov    (%rcx),%r9
  0.00 :        9786:       jne    9ad8 <_dl_lookup_symbol_x+0x408>
  0.00 :        978c:       test   %r9,%r9
  0.00 :        978f:       movq   $0x0,-0xe0(%rbp)
  0.00 :        979a:       je     9ef3 <_dl_lookup_symbol_x+0x823>
  0.00 :        97a0:       mov    -0xe0(%rbp),%rax
  0.00 :        97a7:       mov    -0xd8(%rbp),%r14
  0.00 :        97ae:       jmp    97c4 <_dl_lookup_symbol_x+0xf4>
  0.00 :        97b0:       jne    99a0 <_dl_lookup_symbol_x+0x2d0>
  0.00 :        97b6:       add    $0x8,%r14
  0.00 :        97ba:       xor    %eax,%eax
  0.00 :        97bc:       mov    (%r14),%r9
  0.00 :        97bf:       test   %r9,%r9
  0.00 :        97c2:       je     9810 <_dl_lookup_symbol_x+0x140>
  0.00 :        97c4:       mov    0x18(%rbp),%rsi
  0.00 :        97c8:       mov    0x10(%rbp),%edx
  0.00 :        97cb:       lea    -0xa0(%rbp),%r8
  0.00 :        97d2:       mov    (%rbx),%rcx
  0.00 :        97d5:       mov    -0xb8(%rbp),%rdi
  0.00 :        97dc:       mov    %r15,0x28(%rsp)
  0.00 :        97e1:       mov    %r13d,0x20(%rsp)
  0.00 :        97e6:       mov    %rsi,0x18(%rsp)
  0.00 :        97eb:       mov    -0xd0(%rbp),%rsi
  0.00 :        97f2:       mov    %edx,0x10(%rsp)
  0.00 :        97f6:       lea    -0xb0(%rbp),%rdx
  0.00 :        97fd:       mov    %r12,0x8(%rsp)
  0.00 :        9802:       mov    %rax,(%rsp)
  0.00 :        9806:       callq  8d60 <do_lookup_x>
  0.00 :        980b:       cmp    $0x0,%eax
  0.00 :        980e:       jle    97b0 <_dl_lookup_symbol_x+0xe0>
  0.00 :        9810:       mov    -0xa0(%rbp),%rax
  0.00 :        9817:       test   %rax,%rax
  0.00 :        981a:       je     9ef3 <_dl_lookup_symbol_x+0x823>
  0.00 :        9820:       mov    (%rbx),%rcx
  0.00 :        9823:       test   %rcx,%rcx
  0.00 :        9826:       je     9833 <_dl_lookup_symbol_x+0x163>
  0.00 :        9828:       movzbl 0x5(%rcx),%eax
  0.00 :        982c:       and    $0x3,%eax
  0.00 :        982f:       cmp    $0x3,%al
  0.00 :        9831:       je     9890 <_dl_lookup_symbol_x+0x1c0>
  0.00 :        9833:       mov    -0x98(%rbp),%r14
  0.00 :        983a:       xor    %r9d,%r9d
  0.00 :        983d:       movzbl 0x314(%r14),%eax
  0.00 :        9845:       and    $0x3,%eax
  0.00 :        9848:       cmp    $0x2,%al
  0.00 :        984a:       je     9b1c <_dl_lookup_symbol_x+0x44c>
```

```
 0.00 :        9850:       mov    0x3cc(%r14),%edi
 0.00 :        9857:       test   %edi,%edi
 0.00 :        9859:       je     9b0c <_dl_lookup_symbol_x+0x43c>
 0.00 :        985f:       mov    0x21833b(%rip),%eax        # 221ba0 <_rtld_global_ro>
 0.00 :        9865:       test   $0x804,%eax
 0.00 :        986a:       jne    9b98 <_dl_lookup_symbol_x+0x4c8>
 0.00 :        9870:       mov    -0xa0(%rbp),%rax
 0.00 :        9877:       mov    %rax,(%rbx)
 0.00 :        987a:       lea    -0x28(%rbp),%rsp
 0.00 :        987e:       mov    %r14,%rax
 0.00 :        9881:       pop    %rbx
 0.00 :        9882:       pop    %r12
 0.00 :        9884:       pop    %r13
 0.00 :        9886:       pop    %r14
 0.00 :        9888:       pop    %r15
 0.00 :        988a:       pop    %rbp
 0.00 :        988b:       retq
 0.00 :        988c:       nopl   0x0(%rax)
 0.00 :        9890:       cmp    $0x1,%r13d
 0.00 :        9894:       je     9978 <_dl_lookup_symbol_x+0x2a8>
 0.00 :        989a:       mov    -0xd8(%rbp),%rax
 0.00 :        98a1:       movq   $0x0,-0x80(%rbp)
 0.00 :        98a9:       movq   $0x0,-0x78(%rbp)
 0.00 :        98b1:       mov    (%rax),%r9
 0.00 :        98b4:       test   %r9,%r9
 0.00 :        98b7:       je     9b00 <_dl_lookup_symbol_x+0x430>
 0.00 :        98bd:       mov    -0xe0(%rbp),%rax
 0.00 :        98c4:       mov    -0xd8(%rbp),%r14
 0.00 :        98cb:       jmp    98e1 <_dl_lookup_symbol_x+0x211>
 0.00 :        98cd:       nopl   (%rax)
 0.00 :        98d0:       add    $0x8,%r14
 0.00 :        98d4:       mov    (%r14),%r9
 0.00 :        98d7:       test   %r9,%r9
 0.00 :        98da:       je     992d <_dl_lookup_symbol_x+0x25d>
 0.00 :        98dc:       mov    (%rbx),%rcx
 0.00 :        98df:       xor    %eax,%eax
 0.00 :        98e1:       mov    0x10(%rbp),%esi
 0.00 :        98e4:       mov    0x18(%rbp),%rdx
 0.00 :        98e8:       lea    -0x80(%rbp),%r8
 0.00 :        98ec:       mov    -0xb8(%rbp),%rdi
 0.00 :        98f3:       movq   $0x0,0x28(%rsp)
 0.00 :        98fc:       movl   $0x1,0x20(%rsp)
 0.00 :        9904:       mov    %r12,0x8(%rsp)
 0.00 :        9909:       mov    %esi,0x10(%rsp)
 0.00 :        990d:       mov    -0xd0(%rbp),%rsi
 0.00 :        9914:       mov    %rdx,0x18(%rsp)
 0.00 :        9919:       lea    -0xb0(%rbp),%rdx
 0.00 :        9920:       mov    %rax,(%rsp)
 0.00 :        9924:       callq  8d60 <do_lookup_x>
 0.00 :        9929:       test   %eax,%eax
 0.00 :        992b:       je     98d0 <_dl_lookup_symbol_x+0x200>
 0.00 :        992d:       mov    -0x80(%rbp),%rax
 0.00 :        9931:       test   %rax,%rax
 0.00 :        9934:       je     9b00 <_dl_lookup_symbol_x+0x430>
 0.00 :        993a:       cmp    %r15,-0x78(%rbp)
 0.00 :        993e:       je     9b00 <_dl_lookup_symbol_x+0x430>
 0.00 :        9944:       mov    (%rbx),%rax
 0.00 :        9947:       mov    %r15,-0x98(%rbp)
 0.00 :        994e:       mov    %r15,%r14
 0.00 :        9951:       mov    %rax,-0xa0(%rbp)
 0.00 :        9958:       mov    $0x1,%r9d
 0.00 :        995e:       jmpq   983d <_dl_lookup_symbol_x+0x16d>
 0.00 :        9963:       nopl   0x0(%rax,%rax,1)
 0.00 :        9968:       movq   $0x1505,-0xd0(%rbp)
```

```
 0.00 :        9973:       jmpq   973a <_dl_lookup_symbol_x+0x6a>
 0.00 :        9978:       mov    -0x98(%rbp),%r14
 0.00 :        997f:       cmp    %r15,%r14
 0.00 :        9982:       je     9958 <_dl_lookup_symbol_x+0x288>
 0.00 :        9984:       mov    %rcx,-0xa0(%rbp)
 0.00 :        998b:       mov    %r15,-0x98(%rbp)
 0.00 :        9992:       mov    %r15,%r14
 0.00 :        9995:       mov    $0x1,%r9d
 0.00 :        999b:       jmpq   983d <_dl_lookup_symbol_x+0x16d>
 0.00 :        99a0:       cmpq   $0x0,0x18(%rbp)
 0.00 :        99a5:       jne    97b6 <_dl_lookup_symbol_x+0xe6>
 0.00 :        99ab:       test   %r12,%r12
 0.00 :        99ae:       je     a00e <_dl_lookup_symbol_x+0x93e>
 0.00 :        99b4:       test   %r15,%r15
 0.00 :        99b7:       je     a006 <_dl_lookup_symbol_x+0x936>
 0.00 :        99bd:       mov    0x8(%r15),%r13
 0.00 :        99c1:       lea    0x1193f(%rip),%rdx        # 1b307 <__PRETTY_FUNCTION__.9569+0x39>
 0.00 :        99c8:       mov    -0xb8(%rbp),%rsi
 0.00 :        99cf:       lea    0x11929(%rip),%rcx        # 1b2ff <__PRETTY_FUNCTION__.9569+0x31>
 0.00 :        99d6:       cmp    $0xfffffffe,%eax
 0.00 :        99d9:       lea    0x1152a(%rip),%rax        # 1af0a <intel_02_known+0x4ea>
 0.00 :        99e0:       lea    -0x30(%rbp),%r15
 0.00 :        99e4:       mov    %rdx,-0x60(%rbp)
 0.00 :        99e8:       mov    (%r12),%rdx
 0.00 :        99ec:       mov    %rcx,-0x70(%rbp)
 0.00 :        99f0:       mov    %rsi,-0x68(%rbp)
 0.00 :        99f4:       lea    0x1193f(%rip),%rcx        # 1b33a <__PRETTY_FUNCTION__.9569+0x6c>
 0.00 :        99fb:       lea    0x1194e(%rip),%rsi        # 1b350 <__PRETTY_FUNCTION__.9569+0x82>
 0.00 :        9a02:       mov    %rdx,-0x58(%rbp)
 0.00 :        9a06:       mov    0x10(%r12),%rdx
 0.00 :        9a0b:       lea    -0x68(%rbp),%r12
 0.00 :        9a0f:       mov    %rcx,-0x50(%rbp)
 0.00 :        9a13:       mov    %rsi,-0x40(%rbp)
 0.00 :        9a17:       mov    %r12,%r14
 0.00 :        9a1a:       mov    %rdx,-0x48(%rbp)
 0.00 :        9a1e:       lea    0x118c4(%rip),%rdx        # 1b2e9 <__PRETTY_FUNCTION__.9569+0x1b>
 0.00 :        9a25:       cmovne %rax,%rdx
 0.00 :        9a29:       mov    $0x7,%eax
 0.00 :        9a2e:       mov    %rdx,-0x38(%rbp)
 0.00 :        9a32:       mov    $0x1,%edx
 0.00 :        9a37:       jmp    9a5a <_dl_lookup_symbol_x+0x38a>
 0.00 :        9a39:       nopl   0x0(%rax)
 0.00 :        9a40:       mov    (%r14),%rdi
 0.00 :        9a43:       mov    %rdx,-0x100(%rbp)
 0.00 :        9a4a:       add    $0x8,%r14
 0.00 :        9a4e:       callq  17810 <strlen>
 0.00 :        9a53:       mov    -0x100(%rbp),%rdx
 0.00 :        9a5a:       add    %rax,%rdx
 0.00 :        9a5d:       cmp    %r15,%r14
 0.00 :        9a60:       jne    9a40 <_dl_lookup_symbol_x+0x370>
 0.00 :        9a62:       add    $0x1e,%rdx
 0.00 :        9a66:       lea    0x11892(%rip),%rsi        # 1b2ff <__PRETTY_FUNCTION__.9569+0x31>
 0.00 :        9a6d:       and    $0xfffffffffffffff0,%rdx
 0.00 :        9a71:       sub    %rdx,%rsp
 0.00 :        9a74:       lea    0x3f(%rsp),%r14
 0.00 :        9a79:       and    $0xfffffffffffffff0,%r14
 0.00 :        9a7d:       mov    %r14,%rax
 0.00 :        9a80:       jmp    9a90 <_dl_lookup_symbol_x+0x3c0>
 0.00 :        9a82:       nopw   0x0(%rax,%rax,1)
 0.00 :        9a88:       mov    (%r12),%rsi
 0.00 :        9a8c:       add    $0x8,%r12
 0.00 :        9a90:       mov    %rax,%rdi
 0.00 :        9a93:       callq  18980 <__stpcpy>
 0.00 :        9a98:       cmp    %r15,%r12
```

```
 0.00 :        9a9b:       jne    9a88 <_dl_lookup_symbol_x+0x3b8>
 0.00 :        9a9d:       cmpb   $0x0,0x0(%r13)
 0.00 :        9aa2:       jne    9abc <_dl_lookup_symbol_x+0x3ec>
 0.00 :        9aa4:       mov    0x21822d(%rip),%rax        # 221cd8 <_dl_argv>
 0.00 :        9aab:       mov    (%rax),%r13
 0.00 :        9aae:       lea    0x11199(%rip),%rax        # 1ac4e <intel_02_known+0x22e>
 0.00 :        9ab5:       test   %r13,%r13
 0.00 :        9ab8:       cmove  %rax,%r13
 0.00 :        9abc:       lea    0x118a7(%rip),%rdx        # 1b36a <__PRETTY_FUNCTION__.9569+0x9c>
 0.00 :        9ac3:       mov    %r14,%rcx
 0.00 :        9ac6:       mov    %r13,%rsi
 0.00 :        9ac9:       xor    %edi,%edi
 0.00 :        9acb:       callq  e710 <_dl_signal_cerror>
 0.00 :        9ad0:       jmpq   9f0d <_dl_lookup_symbol_x+0x83d>
 0.00 :        9ad5:       nopl   (%rax)
 0.00 :        9ad8:       mov    (%r9),%rax
 0.00 :        9adb:       mov    0x18(%rbp),%rsi
 0.00 :        9adf:       xor    %edx,%edx
 0.00 :        9ae1:       cmp    %rsi,(%rax)
 0.00 :        9ae4:       je     9af4 <_dl_lookup_symbol_x+0x424>
 0.00 :        9ae6:       add    $0x1,%rdx
 0.00 :        9aea:       mov    0x18(%rbp),%rcx
 0.00 :        9aee:       cmp    %rcx,(%rax,%rdx,8)
 0.00 :        9af2:       jne    9ae6 <_dl_lookup_symbol_x+0x416>
 0.00 :        9af4:       mov    %rdx,-0xe0(%rbp)
 0.00 :        9afb:       jmpq   97a0 <_dl_lookup_symbol_x+0xd0>
 0.00 :        9b00:       mov    -0x98(%rbp),%r14
 0.00 :        9b07:       jmpq   9958 <_dl_lookup_symbol_x+0x288>
 0.00 :        9b0c:       movl   $0x1,0x3cc(%r14)
 0.00 :        9b17:       jmpq   985f <_dl_lookup_symbol_x+0x18f>
 0.00 :        9b1c:       testb  $0x1,0x10(%rbp)
 0.00 :        9b20:       je     9850 <_dl_lookup_symbol_x+0x180>
 0.00 :        9b26:       cmp    %r14,%r15
 0.00 :        9b29:       je     a057 <_dl_lookup_symbol_x+0x987>
 0.00 :        9b2f:       testb  $0x8,0x3d4(%r14)
 0.00 :        9b37:       jne    9850 <_dl_lookup_symbol_x+0x180>
 0.00 :        9b3d:       mov    0x3c0(%r15),%r8
 0.00 :        9b44:       mov    0x3b8(%r15),%rax
 0.00 :        9b4b:       test   %rax,%rax
 0.00 :        9b4e:       mov    %rax,-0xe0(%rbp)
 0.00 :        9b55:       je     9fbf <_dl_lookup_symbol_x+0x8ef>
 0.00 :        9b5b:       mov    (%rax),%rdx
 0.00 :        9b5e:       test   %rdx,%rdx
 0.00 :        9b61:       je     9fbf <_dl_lookup_symbol_x+0x8ef>
 0.00 :        9b67:       xor    %eax,%eax
 0.00 :        9b69:       cmp    %rdx,%r14
 0.00 :        9b6c:       je     9b8c <_dl_lookup_symbol_x+0x4bc>
 0.00 :        9b6e:       mov    -0xe0(%rbp),%rcx
 0.00 :        9b75:       add    $0x1,%eax
 0.00 :        9b78:       mov    %eax,%edx
 0.00 :        9b7a:       mov    (%rcx,%rdx,8),%rdx
 0.00 :        9b7e:       test   %rdx,%rdx
 0.00 :        9b81:       je     9fbf <_dl_lookup_symbol_x+0x8ef>
 0.00 :        9b87:       cmp    %r14,%rdx
 0.00 :        9b8a:       jne    9b6e <_dl_lookup_symbol_x+0x49e>
 0.00 :        9b8c:       mov    -0x98(%rbp),%r14
 0.00 :        9b93:       jmpq   9850 <_dl_lookup_symbol_x+0x180>
 0.00 :        9b98:       test   $0x4,%al
 0.00 :        9b9a:       mov    0x8(%r15),%rsi
 0.00 :        9b9e:       je     9c2e <_dl_lookup_symbol_x+0x55e>
 0.00 :        9ba4:       mov    0x8(%r14),%rcx
 0.00 :        9ba8:       lea    0x11763(%rip),%rdi        # 1b312 <__PRETTY_FUNCTION__.9569+0x44>
 0.00 :        9baf:       lea    0x11766(%rip),%rax        # 1b31c <__PRETTY_FUNCTION__.9569+0x4e>
 0.00 :        9bb6:       test   %r9d,%r9d
```

```
 0.00 :        9bb9:       mov    0x30(%r14),%r8
 0.00 :        9bbd:       cmove  %rax,%rdi
 0.00 :        9bc1:       cmpb   $0x0,(%rcx)
 0.00 :        9bc4:       jne    9bd0 <_dl_lookup_symbol_x+0x500>
 0.00 :        9bc6:       mov    0x21810b(%rip),%rax        # 221cd8 <_dl_argv>
 0.00 :        9bcd:       mov    (%rax),%rcx
 0.00 :        9bd0:       cmpb   $0x0,(%rsi)
 0.00 :        9bd3:       mov    0x30(%r15),%rdx
 0.00 :        9bd7:       jne    9bf1 <_dl_lookup_symbol_x+0x521>
 0.00 :        9bd9:       mov    0x2180f8(%rip),%rax        # 221cd8 <_dl_argv>
 0.00 :        9be0:       mov    (%rax),%rsi
 0.00 :        9be3:       lea    0x11064(%rip),%rax        # 1ac4e <intel_02_known+0x22e>
 0.00 :        9bea:       test   %rsi,%rsi
 0.00 :        9bed:       cmove  %rax,%rsi
 0.00 :        9bf1:       mov    -0xb8(%rbp),%rax
 0.00 :        9bf8:       mov    %rdi,%r9
 0.00 :        9bfb:       lea    0x1364e(%rip),%rdi        # 1d250 <__PRETTY_FUNCTION__.4816+0x16e7>
 0.00 :        9c02:       mov    %rax,(%rsp)
 0.00 :        9c06:       xor    %eax,%eax
 0.00 :        9c08:       callq  f8f0 <_dl_debug_printf>
 0.00 :        9c0d:       test   %r12,%r12
 0.00 :        9c10:       je     9eda <_dl_lookup_symbol_x+0x80a>
 0.00 :        9c16:       mov    (%r12),%rsi
 0.00 :        9c1a:       lea    0x1176e(%rip),%rdi        # 1b38f <__PRETTY_FUNCTION__.9569+0xc1>
 0.00 :        9c21:       xor    %eax,%eax
 0.00 :        9c23:       callq  f950 <_dl_debug_printf_c>
 0.00 :        9c28:       mov    0x217f72(%rip),%eax        # 221ba0 <_rtld_global_ro>
 0.00 :        9c2e:       test   $0x8,%ah
 0.00 :        9c31:       je     9ece <_dl_lookup_symbol_x+0x7fe>
 0.00 :        9c37:       mov    0x218002(%rip),%rdx        # 221c40 <_rtld_global_ro+0xa0>
 0.00 :        9c3e:       movq   $0x0,-0x90(%rbp)
 0.00 :        9c49:       movq   $0x0,-0x88(%rbp)
 0.00 :        9c54:       mov    0x2183a5(%rip),%rax        # 222000 <_rtld_global>
 0.00 :        9c5b:       test   %rdx,%rdx
 0.00 :        9c5e:       je     9df0 <_dl_lookup_symbol_x+0x720>
 0.00 :        9c64:       cmp    %rax,%rdx
 0.00 :        9c67:       je     9df0 <_dl_lookup_symbol_x+0x720>
 0.00 :        9c6d:       mov    -0xa0(%rbp),%rax
 0.00 :        9c74:       xor    %edx,%edx
 0.00 :        9c76:       test   %rax,%rax
 0.00 :        9c79:       je     9c94 <_dl_lookup_symbol_x+0x5c4>
 0.00 :        9c7b:       movzbl 0x4(%rax),%ecx
 0.00 :        9c7f:       and    $0xf,%ecx
 0.00 :        9c82:       cmp    $0x6,%cl
 0.00 :        9c85:       je     9de5 <_dl_lookup_symbol_x+0x715>
 0.00 :        9c8b:       cmp    $0xa,%cl
 0.00 :        9c8e:       jne    9c94 <_dl_lookup_symbol_x+0x5c4>
 0.00 :        9c90:       or     $0x8,%r13d
 0.00 :        9c94:       test   %edx,%edx
 0.00 :        9c96:       jne    9cb3 <_dl_lookup_symbol_x+0x5e3>
 0.00 :        9c98:       mov    0x217fa1(%rip),%rcx        # 221c40 <_rtld_global_ro+0xa0>
 0.00 :        9c9f:       cmp    %rcx,%r15
 0.00 :        9ca2:       je     9cb3 <_dl_lookup_symbol_x+0x5e3>
 0.00 :        9ca4:       test   %rcx,%rcx
 0.00 :        9ca7:       je     9cb3 <_dl_lookup_symbol_x+0x5e3>
 0.00 :        9ca9:       cmp    $0x3,%r13d
 0.00 :        9cad:       jle    9d8b <_dl_lookup_symbol_x+0x6bb>
 0.00 :        9cb3:       test   %rax,%rax
 0.00 :        9cb6:       je     9d97 <_dl_lookup_symbol_x+0x6c7>
 0.00 :        9cbc:       mov    0x8(%rax),%rsi
 0.00 :        9cc0:       mov    -0x98(%rbp),%rax
 0.00 :        9cc7:       mov    0x340(%rax),%rcx
 0.00 :        9cce:       mov    0x340(%r15),%r8
 0.00 :        9cd5:       mov    (%rbx),%rax
```

```
 0.00 :        9cd8:       sub    %r8,%rax
 0.00 :        9cdb:       test   %edx,%edx
 0.00 :        9cdd:       je     9da0 <_dl_lookup_symbol_x+0x6d0>
 0.00 :        9ce3:       mov    %rsi,0x20(%rsp)
 0.00 :        9ce8:       lea    0x116c9(%rip),%rdx        # 1b3b8 <__PRETTY_FUNCTION__.9569+0xea>
 0.00 :        9cef:       lea    0x13592(%rip),%rsi        # 1d288 <__PRETTY_FUNCTION__.4816+0x171f>
 0.00 :        9cf6:       mov    %rcx,0x10(%rsp)
 0.00 :        9cfb:       mov    %rax,(%rsp)
 0.00 :        9cff:       mov    $0x10,%r9d
 0.00 :        9d05:       xor    %eax,%eax
 0.00 :        9d07:       mov    $0x10,%ecx
 0.00 :        9d0c:       movl   $0x10,0x18(%rsp)
 0.00 :        9d14:       movl   $0x10,0x8(%rsp)
 0.00 :        9d1c:       mov    $0x1,%edi
 0.00 :        9d21:       callq  f9b0 <_dl_dprintf>
 0.00 :        9d26:       mov    -0x90(%rbp),%rax
 0.00 :        9d2d:       xor    %r9d,%r9d
 0.00 :        9d30:       xor    %ecx,%ecx
 0.00 :        9d32:       test   %rax,%rax
 0.00 :        9d35:       je     9d49 <_dl_lookup_symbol_x+0x679>
 0.00 :        9d37:       mov    0x8(%rax),%r9
 0.00 :        9d3b:       mov    -0x88(%rbp),%rax
 0.00 :        9d42:       mov    0x340(%rax),%rcx
 0.00 :        9d49:       lea    0x11646(%rip),%rsi        # 1b396 <__PRETTY_FUNCTION__.9569+0xc8>
 0.00 :        9d50:       mov    $0x10,%r8d
 0.00 :        9d56:       mov    $0x10,%edx
 0.00 :        9d5b:       mov    $0x1,%edi
 0.00 :        9d60:       xor    %eax,%eax
 0.00 :        9d62:       callq  f9b0 <_dl_dprintf>
 0.00 :        9d67:       mov    -0xb8(%rbp),%rcx
 0.00 :        9d6e:       lea    0x11634(%rip),%rsi        # 1b3a9 <__PRETTY_FUNCTION__.9569+0xdb>
 0.00 :        9d75:       xor    %eax,%eax
 0.00 :        9d77:       mov    %r13d,%edx
 0.00 :        9d7a:       mov    $0x1,%edi
 0.00 :        9d7f:       callq  f9b0 <_dl_dprintf>
 0.00 :        9d84:       mov    -0xa0(%rbp),%rax
 0.00 :        9d8b:       mov    -0x98(%rbp),%r14
 0.00 :        9d92:       jmpq   9877 <_dl_lookup_symbol_x+0x1a7>
 0.00 :        9d97:       xor    %esi,%esi
 0.00 :        9d99:       xor    %ecx,%ecx
 0.00 :        9d9b:       jmpq   9cce <_dl_lookup_symbol_x+0x5fe>
 0.00 :        9da0:       mov    %rsi,0x20(%rsp)
 0.00 :        9da5:       lea    0x11605(%rip),%rdx        # 1b3b1 <__PRETTY_FUNCTION__.9569+0xe3>
 0.00 :        9dac:       lea    0x134d5(%rip),%rsi        # 1d288 <__PRETTY_FUNCTION__.4816+0x171f>
 0.00 :        9db3:       mov    %rcx,0x10(%rsp)
 0.00 :        9db8:       mov    %rax,(%rsp)
 0.00 :        9dbc:       mov    $0x10,%r9d
 0.00 :        9dc2:       movl   $0x10,0x18(%rsp)
 0.00 :        9dca:       movl   $0x10,0x8(%rsp)
 0.00 :        9dd2:       mov    $0x10,%ecx
 0.00 :        9dd7:       mov    $0x1,%edi
 0.00 :        9ddc:       xor    %eax,%eax
 0.00 :        9dde:       callq  f9b0 <_dl_dprintf>
 0.00 :        9de3:       jmp    9d67 <_dl_lookup_symbol_x+0x697>
 0.00 :        9de5:       mov    $0x4,%r13d
 0.00 :        9deb:       jmpq   9c94 <_dl_lookup_symbol_x+0x5c4>
 0.00 :        9df0:       cmp    %r15,%rax
 0.00 :        9df3:       je     9c6d <_dl_lookup_symbol_x+0x59d>
 0.00 :        9df9:       mov    -0xb8(%rbp),%rdx
 0.00 :        9e00:       mov    $0x1505,%r14d
 0.00 :        9e06:       movzbl (%rdx),%eax
 0.00 :        9e09:       test   %al,%al
 0.00 :        9e0b:       je     9e28 <_dl_lookup_symbol_x+0x758>
 0.00 :        9e0d:       mov    %r14,%rcx
```

```
  0.00 :         9e10:       add    $0x1,%rdx
  0.00 :         9e14:       shl    $0x5,%rcx
  0.00 :         9e18:       add    %rcx,%r14
  0.00 :         9e1b:       add    %rax,%r14
  0.00 :         9e1e:       movzbl (%rdx),%eax
  0.00 :         9e21:       test   %al,%al
  0.00 :         9e23:       jne    9e0d <_dl_lookup_symbol_x+0x73d>
  0.00 :         9e25:       mov    %r14d,%r14d
  0.00 :         9e28:       mov    0x218219(%rip),%rcx        # 222048 <_rtld_global+0x48>
  0.00 :         9e2f:       mov    %r15,0x28(%rsp)
  0.00 :         9e34:       mov    $0xffffffff,%esi
  0.00 :         9e39:       movq   $0x0,0x218204(%rip)        # 222048 <_rtld_global+0x48>
  0.00 :         9e44:       mov    %r13d,0x20(%rsp)
  0.00 :         9e49:       lea    -0xa8(%rbp),%rdx
  0.00 :         9e50:       movq   $0x0,0x18(%rsp)
  0.00 :         9e59:       movl   $0x0,0x10(%rsp)
  0.00 :         9e61:       lea    -0x90(%rbp),%r8
  0.00 :         9e68:       mov    %rcx,-0xc8(%rbp)
  0.00 :         9e6f:       mov    %r12,0x8(%rsp)
  0.00 :         9e74:       movq   $0x0,(%rsp)
  0.00 :         9e7c:       mov    (%rbx),%rcx
  0.00 :         9e7f:       mov    0x388(%r15),%r9
  0.00 :         9e86:       mov    -0xb8(%rbp),%rdi
  0.00 :         9e8d:       mov    %rsi,-0xa8(%rbp)
  0.00 :         9e94:       mov    %r14,%rsi
  0.00 :         9e97:       callq  8d60 <do_lookup_x>
  0.00 :         9e9c:       mov    -0x90(%rbp),%rax
  0.00 :         9ea3:       mov    -0xa0(%rbp),%rdx
  0.00 :         9eaa:       cmp    %rdx,%rax
  0.00 :         9ead:       je     a02d <_dl_lookup_symbol_x+0x95d>
  0.00 :         9eb3:       mov    %rdx,%rax
  0.00 :         9eb6:       mov    $0x1,%edx
  0.00 :         9ebb:       mov    -0xc8(%rbp),%rcx
  0.00 :         9ec2:       mov    %rcx,0x21817f(%rip)        # 222048 <_rtld_global+0x48>
  0.00 :         9ec9:       jmpq   9c76 <_dl_lookup_symbol_x+0x5a6>
  0.00 :         9ece:       mov    -0x98(%rbp),%r14
  0.00 :         9ed5:       jmpq   9870 <_dl_lookup_symbol_x+0x1a0>
  0.00 :         9eda:       lea    0x11028(%rip),%rdi        # 1af09 <intel_02_known+0x4e9>
  0.00 :         9ee1:       xor    %eax,%eax
  0.00 :         9ee3:       callq  f950 <_dl_debug_printf_c>
  0.00 :         9ee8:       mov    0x217cb2(%rip),%eax        # 221ba0 <_rtld_global_ro>
  0.00 :         9eee:       jmpq   9c2e <_dl_lookup_symbol_x+0x55e>
  0.00 :         9ef3:       mov    (%rbx),%rax
  0.00 :         9ef6:       test   %rax,%rax
  0.00 :         9ef9:       je     9f06 <_dl_lookup_symbol_x+0x836>
  0.00 :         9efb:       movzbl 0x4(%rax),%eax
  0.00 :         9eff:       shr    $0x4,%al
  0.00 :         9f02:       cmp    $0x2,%al
  0.00 :         9f04:       je     9f0d <_dl_lookup_symbol_x+0x83d>
  0.00 :         9f06:       cmpq   $0x0,0x18(%rbp)
  0.00 :         9f0b:       je     9f1c <_dl_lookup_symbol_x+0x84c>
  0.00 :         9f0d:       movq   $0x0,(%rbx)
  0.00 :         9f14:       xor    %r14d,%r14d
  0.00 :         9f17:       jmpq   987a <_dl_lookup_symbol_x+0x1aa>
  0.00 :         9f1c:       testb  $0x1,0x217c7e(%rip)        # 221ba1 <_rtld_global_ro+0x1>
  0.00 :         9f23:       jne    9f0d <_dl_lookup_symbol_x+0x83d>
  0.00 :         9f25:       test   %r15,%r15
  0.00 :         9f28:       lea    0x10fdb(%rip),%r13        # 1af0a <intel_02_known+0x4ea>
  0.00 :         9f2f:       je     9f35 <_dl_lookup_symbol_x+0x865>
  0.00 :         9f31:       mov    0x8(%r15),%r13
  0.00 :         9f35:       test   %r12,%r12
  0.00 :         9f38:       je     a1a5 <_dl_lookup_symbol_x+0xad5>
  0.00 :         9f3e:       mov    (%r12),%rax
  0.00 :         9f42:       lea    0x10fc1(%rip),%rcx        # 1af0a <intel_02_known+0x4ea>
```

```
 0.00 :        9f49:       lea    0x113b7(%rip),%rdx        # 1b307 <__PRETTY_FUNCTION__.9569+0x39>
 0.00 :        9f50:       test   %rax,%rax
 0.00 :        9f53:       cmove  %rcx,%rax
 0.00 :        9f57:       mov    -0xb8(%rbp),%rsi
 0.00 :        9f5e:       lea    0x140eb(%rip),%rcx        # 1e050 <undefined_msg>
 0.00 :        9f65:       mov    %rax,-0x58(%rbp)
 0.00 :        9f69:       mov    %rdx,-0x60(%rbp)
 0.00 :        9f6d:       mov    $0x12,%eax
 0.00 :        9f72:       xor    %r12d,%r12d
 0.00 :        9f75:       mov    %rcx,-0x70(%rbp)
 0.00 :        9f79:       mov    $0x1,%r14d
 0.00 :        9f7f:       mov    %rsi,-0x68(%rbp)
 0.00 :        9f83:       add    $0x1,%r12
 0.00 :        9f87:       add    %rax,%r14
 0.00 :        9f8a:       cmp    $0x4,%r12
 0.00 :        9f8e:       je     a16b <_dl_lookup_symbol_x+0xa9b>
 0.00 :        9f94:       mov    -0x70(%rbp,%r12,8),%rdi
 0.00 :        9f99:       callq  17810 <strlen>
 0.00 :        9f9e:       jmp    9f83 <_dl_lookup_symbol_x+0x8b3>
 0.00 :        9fa0:       lea    0x14089(%rip),%rcx        # 1e030 <__PRETTY_FUNCTION__.9516>
 0.00 :        9fa7:       lea    0x1132f(%rip),%rsi        # 1b2dd <__PRETTY_FUNCTION__.9569+0xf>
 0.00 :        9fae:       lea    0x131fb(%rip),%rdi        # 1d1b0 <__PRETTY_FUNCTION__.4816+0x1647>
 0.00 :        9fb5:       mov    $0x2d8,%edx
 0.00 :        9fba:       callq  15980 <__GI___assert_fail>
 0.00 :        9fbf:       test   %r8,%r8
 0.00 :        9fc2:       je     a1e2 <_dl_lookup_symbol_x+0xb12>
 0.00 :        9fc8:       mov    (%r8),%esi
 0.00 :        9fcb:       test   %esi,%esi
 0.00 :        9fcd:       mov    %esi,-0xd0(%rbp)
 0.00 :        9fd3:       je     a1ec <_dl_lookup_symbol_x+0xb1c>
 0.00 :        9fd9:       cmp    0x8(%r8),%r14
 0.00 :        9fdd:       je     9b8c <_dl_lookup_symbol_x+0x4bc>
 0.00 :        9fe3:       mov    %r8,%rdx
 0.00 :        9fe6:       xor    %eax,%eax
 0.00 :        9fe8:       add    $0x1,%eax
 0.00 :        9feb:       cmp    -0xd0(%rbp),%eax
 0.00 :        9ff1:       je     a1ec <_dl_lookup_symbol_x+0xb1c>
 0.00 :        9ff7:       add    $0x8,%rdx
 0.00 :        9ffb:       cmp    %r14,0x8(%rdx)
 0.00 :        9fff:       jne    9fe8 <_dl_lookup_symbol_x+0x918>
 0.00 :        a001:       jmpq   9b8c <_dl_lookup_symbol_x+0x4bc>
 0.00 :        a006:       xor    %r13d,%r13d
 0.00 :        a009:       jmpq   99c1 <_dl_lookup_symbol_x+0x2f1>
 0.00 :        a00e:       lea    0x1401b(%rip),%rcx        # 1e030 <__PRETTY_FUNCTION__.9516>
 0.00 :        a015:       lea    0x112c1(%rip),%rsi        # 1b2dd <__PRETTY_FUNCTION__.9569+0xf>
 0.00 :        a01c:       lea    0x11300(%rip),%rdi        # 1b323 <__PRETTY_FUNCTION__.9569+0x55>
 0.00 :        a023:       mov    $0x2ee,%edx
 0.00 :        a028:       callq  15980 <__GI___assert_fail>
 0.00 :        a02d:       mov    -0x88(%rbp),%rsi
 0.00 :        a034:       cmp    -0x98(%rbp),%rsi
 0.00 :        a03b:       mov    $0x1,%edx
 0.00 :        a040:       jne    9ebb <_dl_lookup_symbol_x+0x7eb>
 0.00 :        a046:       cmpb   $0x0,0x315(%r15)
 0.00 :        a04e:       js     a05f <_dl_lookup_symbol_x+0x98f>
 0.00 :        a050:       xor    %edx,%edx
 0.00 :        a052:       jmpq   9ebb <_dl_lookup_symbol_x+0x7eb>
 0.00 :        a057:       mov    %r15,%r14
 0.00 :        a05a:       jmpq   9850 <_dl_lookup_symbol_x+0x180>
 0.00 :        a05f:       test   %rax,%rax
 0.00 :        a062:       je     a050 <_dl_lookup_symbol_x+0x980>
 0.00 :        a064:       movzbl 0x4(%rax),%edx
 0.00 :        a068:       shr    $0x4,%dl
 0.00 :        a06b:       cmp    $0xa,%dl
 0.00 :        a06e:       jne    a050 <_dl_lookup_symbol_x+0x980>
```

```
 0.00 :         a070:       mov    0x388(%r15),%r10
 0.00 :         a077:       movq   $0x0,-0x80(%rbp)
 0.00 :         a07f:       movq   $0x0,-0x78(%rbp)
 0.00 :         a087:       mov    0x8(%r10),%edx
 0.00 :         a08b:       test   %rdx,%rdx
 0.00 :         a08e:       je     a050 <_dl_lookup_symbol_x+0x980>
 0.00 :         a090:       mov    (%r10),%rcx
 0.00 :         a093:       cmp    (%rcx),%rsi
 0.00 :         a096:       je     a3c9 <_dl_lookup_symbol_x+0xcf9>
 0.00 :         a09c:       xor    %edi,%edi
 0.00 :         a09e:       add    $0x1,%rdi
 0.00 :         a0a2:       cmp    %rdx,%rdi
 0.00 :         a0a5:       jae    a0ad <_dl_lookup_symbol_x+0x9dd>
 0.00 :         a0a7:       cmp    (%rcx,%rdi,8),%rsi
 0.00 :         a0ab:       jne    a09e <_dl_lookup_symbol_x+0x9ce>
 0.00 :         a0ad:       lea    0x1(%rdi),%r11
 0.00 :         a0b1:       cmp    %rdx,%r11
 0.00 :         a0b4:       jae    a050 <_dl_lookup_symbol_x+0x980>
 0.00 :         a0b6:       lea    0x0(,%r11,8),%rsi
 0.00 :         a0be:       mov    %rsi,-0xc0(%rbp)
 0.00 :         a0c5:       mov    -0xc0(%rbp),%rsi
 0.00 :         a0cc:       mov    (%rcx,%rsi,1),%rax
 0.00 :         a0d0:       cmpq   $0x0,0xc0(%rax)
 0.00 :         a0d8:       je     a14e <_dl_lookup_symbol_x+0xa7e>
 0.00 :         a0da:       mov    (%rbx),%rcx
 0.00 :         a0dd:       mov    -0xb8(%rbp),%rdi
 0.00 :         a0e4:       lea    0x2c8(%rax),%r9
 0.00 :         a0eb:       lea    -0x80(%rbp),%r8
 0.00 :         a0ef:       lea    -0xa8(%rbp),%rdx
 0.00 :         a0f6:       mov    %r15,0x28(%rsp)
 0.00 :         a0fb:       mov    %r13d,0x20(%rsp)
 0.00 :         a100:       movq   $0x0,0x18(%rsp)
 0.00 :         a109:       mov    %r14,%rsi
 0.00 :         a10c:       movl   $0x0,0x10(%rsp)
 0.00 :         a114:       mov    %r12,0x8(%rsp)
 0.00 :         a119:       movq   $0x0,(%rsp)
 0.00 :         a121:       mov    %r10,-0x100(%rbp)
 0.00 :         a128:       mov    %r11,-0x108(%rbp)
 0.00 :         a12f:       callq  8d60 <do_lookup_x>
 0.00 :         a134:       test   %eax,%eax
 0.00 :         a136:       mov    -0x100(%rbp),%r10
 0.00 :         a13d:       mov    -0x108(%rbp),%r11
 0.00 :         a144:       jg     a3d4 <_dl_lookup_symbol_x+0xd04>
 0.00 :         a14a:       mov    0x8(%r10),%edx
 0.00 :         a14e:       add    $0x1,%r11
 0.00 :         a152:       addq   $0x8,-0xc0(%rbp)
 0.00 :         a15a:       cmp    %rdx,%r11
 0.00 :         a15d:       jae    a47a <_dl_lookup_symbol_x+0xdaa>
 0.00 :         a163:       mov    (%r10),%rcx
 0.00 :         a166:       jmpq   a0c5 <_dl_lookup_symbol_x+0x9f5>
 0.00 :         a16b:       add    $0x1e,%r14
 0.00 :         a16f:       lea    0x13eda(%rip),%rsi        # 1e050 <undefined_msg>
 0.00 :         a176:       and    $0xfffffffffffffff0,%r14
 0.00 :         a17a:       sub    %r14,%rsp
 0.00 :         a17d:       xor    %r14d,%r14d
 0.00 :         a180:       lea    0x3f(%rsp),%r12
 0.00 :         a185:       and    $0xfffffffffffffff0,%r12
 0.00 :         a189:       mov    %r12,%rax
 0.00 :         a18c:       mov    %rax,%rdi
 0.00 :         a18f:       add    $0x1,%r14
 0.00 :         a193:       callq  18980 <__stpcpy>
 0.00 :         a198:       cmp    $0x4,%r14
 0.00 :         a19c:       je     a1b4 <_dl_lookup_symbol_x+0xae4>
 0.00 :         a19e:       mov    -0x70(%rbp,%r14,8),%rsi
```

```
 0.00 :        a1a3:       jmp    a18c <_dl_lookup_symbol_x+0xabc>
 0.00 :        a1a5:       lea    0x10d5e(%rip),%rdx        # 1af0a <intel_02_known+0x4ea>
 0.00 :        a1ac:       mov    %rdx,%rax
 0.00 :        a1af:       jmpq   9f57 <_dl_lookup_symbol_x+0x887>
 0.00 :        a1b4:       cmpb   $0x0,0x0(%r13)
 0.00 :        a1b9:       jne    a1d3 <_dl_lookup_symbol_x+0xb03>
 0.00 :        a1bb:       mov    0x217b16(%rip),%rax        # 221cd8 <_dl_argv>
 0.00 :        a1c2:       mov    (%rax),%r13
 0.00 :        a1c5:       lea    0x10a82(%rip),%rax        # 1ac4e <intel_02_known+0x22e>
 0.00 :        a1cc:       test   %r13,%r13
 0.00 :        a1cf:       cmove  %rax,%r13
 0.00 :        a1d3:       mov    %r12,%rcx
 0.00 :        a1d6:       lea    0x1119e(%rip),%rdx        # 1b37b <__PRETTY_FUNCTION__.9569+0xad>
 0.00 :        a1dd:       jmpq   9ac6 <_dl_lookup_symbol_x+0x3f6>
 0.00 :        a1e2:       movl   $0x0,-0xd0(%rbp)
 0.00 :        a1ec:       mov    0x10(%rbp),%edx
 0.00 :        a1ef:       mov    0x460(%r14),%rax
 0.00 :        a1f6:       and    $0x4,%edx
 0.00 :        a1f9:       mov    %rax,-0xe8(%rbp)
 0.00 :        a200:       movslq %edx,%rcx
 0.00 :        a203:       mov    %edx,-0xec(%rbp)
 0.00 :        a209:       test   %rcx,%rcx
 0.00 :        a20c:       mov    %rcx,-0xf8(%rbp)
 0.00 :        a213:       jne    a317 <_dl_lookup_symbol_x+0xc47>
 0.00 :        a219:       mov    %r9d,-0x110(%rbp)
 0.00 :        a220:       lea    0x2186e1(%rip),%rdi        # 222908 <_rtld_global+0x908>
 0.00 :        a227:       callq  *0x218cd3(%rip)        # 222f00 <_rtld_global+0xf00>
 0.00 :        a22d:       mov    -0x110(%rbp),%r9d
 0.00 :        a234:       mov    0x30(%r15),%rax
 0.00 :        a238:       lea    0x217dc1(%rip),%rdx        # 222000 <_rtld_global>
 0.00 :        a23f:       lea    (%rax,%rax,8),%rax
 0.00 :        a243:       shl    $0x4,%rax
 0.00 :        a247:       mov    (%rdx,%rax,1),%rax
 0.00 :        a24b:       cmp    %r14,%rax
 0.00 :        a24e:       je     a25e <_dl_lookup_symbol_x+0xb8e>
 0.00 :        a250:       test   %rax,%rax
 0.00 :        a253:       je     a25e <_dl_lookup_symbol_x+0xb8e>
 0.00 :        a255:       mov    0x18(%rax),%rax
 0.00 :        a259:       cmp    %r14,%rax
 0.00 :        a25c:       jne    a250 <_dl_lookup_symbol_x+0xb80>
 0.00 :        a25e:       test   %rax,%rax
 0.00 :        a261:       mov    $0xffffffff,%edx
 0.00 :        a266:       je     a281 <_dl_lookup_symbol_x+0xbb1>
 0.00 :        a268:       mov    -0xe8(%rbp),%rax
 0.00 :        a26f:       cmp    0x460(%r14),%rax
 0.00 :        a276:       je     a3fb <_dl_lookup_symbol_x+0xd2b>
 0.00 :        a27c:       mov    $0xffffffff,%edx
 0.00 :        a281:       mov    %edx,-0x100(%rbp)
 0.00 :        a287:       mov    %r9d,-0x110(%rbp)
 0.00 :        a28e:       lea    0x218673(%rip),%rdi        # 222908 <_rtld_global+0x908>
 0.00 :        a295:       callq  *0x218c6d(%rip)        # 222f08 <_rtld_global+0xf08>
 0.00 :        a29b:       cmpq   $0x0,-0xf8(%rbp)
 0.00 :        a2a3:       mov    -0x100(%rbp),%edx
 0.00 :        a2a9:       mov    -0x110(%rbp),%r9d
 0.00 :        a2b0:       je     a2be <_dl_lookup_symbol_x+0xbee>
 0.00 :        a2b2:       movl   $0x1,%fs:0x1c
 0.00 :        a2be:       cmp    $0xffffffff,%edx
 0.00 :        a2c1:       jne    9b8c <_dl_lookup_symbol_x+0x4bc>
 0.00 :        a2c7:       mov    -0xec(%rbp),%r8d
 0.00 :        a2ce:       test   %r8d,%r8d
 0.00 :        a2d1:       je     a2e1 <_dl_lookup_symbol_x+0xc11>
 0.00 :        a2d3:       mov    0x380(%r15),%rcx
 0.00 :        a2da:       mov    %rcx,-0xd8(%rbp)
 0.00 :        a2e1:       mov    0x18(%rbp),%rsi
```

```
 0.00 :        a2e5:       mov    0x10(%rbp),%eax
 0.00 :        a2e8:       mov    %r13d,%r9d
 0.00 :        a2eb:       mov    -0xd8(%rbp),%rcx
 0.00 :        a2f2:       mov    -0xb8(%rbp),%rdi
 0.00 :        a2f9:       mov    %r12,%r8
 0.00 :        a2fc:       mov    %rbx,%rdx
 0.00 :        a2ff:       mov    %rsi,0x8(%rsp)
 0.00 :        a304:       mov    %eax,(%rsp)
 0.00 :        a307:       mov    %r15,%rsi
 0.00 :        a30a:       callq  96d0 <_dl_lookup_symbol_x>
 0.00 :        a30f:       mov    %rax,%r14
 0.00 :        a312:       jmpq   987a <_dl_lookup_symbol_x+0x1aa>
 0.00 :        a317:       xor    %eax,%eax
 0.00 :        a319:       xchg   %eax,%fs:0x1c
 0.00 :        a321:       cmp    $0x2,%eax
 0.00 :        a324:       jne    a341 <_dl_lookup_symbol_x+0xc71>
 0.00 :        a326:       mov    %fs:0x10,%rdi
 0.00 :        a32f:       mov    $0x1,%edx
 0.00 :        a334:       add    $0x1c,%rdi
 0.00 :        a338:       mov    $0xca,%al
 0.00 :        a33a:       mov    $0x81,%esi
 0.00 :        a33f:       syscall
 0.00 :        a341:       mov    %r8,-0x108(%rbp)
 0.00 :        a348:       mov    %r9d,-0x110(%rbp)
 0.00 :        a34f:       lea    0x2185b2(%rip),%rdi        # 222908 <_rtld_global+0x908>
 0.00 :        a356:       callq  *0x218ba4(%rip)        # 222f00 <_rtld_global+0xf00>
 0.00 :        a35c:       mov    %r14,%rax
 0.00 :        a35f:       mov    0x3b8(%r15),%rdx
 0.00 :        a366:       mov    -0x108(%rbp),%r8
 0.00 :        a36d:       cmp    %rdx,-0xe0(%rbp)
 0.00 :        a374:       mov    %rax,%r14
 0.00 :        a377:       mov    -0x110(%rbp),%r9d
 0.00 :        a37e:       je     a427 <_dl_lookup_symbol_x+0xd57>
 0.00 :        a384:       test   %rdx,%rdx
 0.00 :        a387:       je     a427 <_dl_lookup_symbol_x+0xd57>
 0.00 :        a38d:       mov    (%rdx),%rcx
 0.00 :        a390:       test   %rcx,%rcx
 0.00 :        a393:       je     a427 <_dl_lookup_symbol_x+0xd57>
 0.00 :        a399:       cmp    %rcx,%rax
 0.00 :        a39c:       je     a3b3 <_dl_lookup_symbol_x+0xce3>
 0.00 :        a39e:       xor    %ecx,%ecx
 0.00 :        a3a0:       add    $0x1,%ecx
 0.00 :        a3a3:       mov    %ecx,%esi
 0.00 :        a3a5:       mov    (%rdx,%rsi,8),%rsi
 0.00 :        a3a9:       test   %rsi,%rsi
 0.00 :        a3ac:       je     a427 <_dl_lookup_symbol_x+0xd57>
 0.00 :        a3ae:       cmp    %rsi,%rax
 0.00 :        a3b1:       jne    a3a0 <_dl_lookup_symbol_x+0xcd0>
 0.00 :        a3b3:       mov    -0xe8(%rbp),%rsi
 0.00 :        a3ba:       cmp    0x460(%rax),%rsi
 0.00 :        a3c1:       jne    a27c <_dl_lookup_symbol_x+0xbac>
 0.00 :        a3c7:       jmp    a420 <_dl_lookup_symbol_x+0xd50>
 0.00 :        a3c9:       mov    $0x1,%r11d
 0.00 :        a3cf:       jmpq   a0b1 <_dl_lookup_symbol_x+0x9e1>
 0.00 :        a3d4:       mov    -0x80(%rbp),%rax
 0.00 :        a3d8:       mov    $0x1,%edx
 0.00 :        a3dd:       mov    %rax,-0x90(%rbp)
 0.00 :        a3e4:       mov    -0x78(%rbp),%rax
 0.00 :        a3e8:       mov    %rax,-0x88(%rbp)
 0.00 :        a3ef:       mov    -0xa0(%rbp),%rax
 0.00 :        a3f6:       jmpq   9ebb <_dl_lookup_symbol_x+0x7eb>
 0.00 :        a3fb:       mov    0x3d4(%r14),%edx
 0.00 :        a402:       test   $0x8,%dl
 0.00 :        a405:       jne    a420 <_dl_lookup_symbol_x+0xd50>
```

```
 0.00 :        a407:       movzbl 0x314(%r15),%eax
 0.00 :        a40f:       and    $0x3,%eax
 0.00 :        a412:       cmp    $0x2,%al
 0.00 :        a414:       je     a488 <_dl_lookup_symbol_x+0xdb8>
 0.00 :        a416:       or     $0x8,%edx
 0.00 :        a419:       mov    %edx,0x3d4(%r14)
 0.00 :        a420:       xor    %edx,%edx
 0.00 :        a422:       jmpq   a281 <_dl_lookup_symbol_x+0xbb1>
 0.00 :        a427:       mov    0x3c0(%r15),%rdx
 0.00 :        a42e:       test   %rdx,%rdx
 0.00 :        a431:       je     a234 <_dl_lookup_symbol_x+0xb64>
 0.00 :        a437:       cmp    %rdx,%r8
 0.00 :        a43a:       je     a668 <_dl_lookup_symbol_x+0xf98>
 0.00 :        a440:       mov    (%rdx),%esi
 0.00 :        a442:       test   %esi,%esi
 0.00 :        a444:       mov    %esi,-0xd0(%rbp)
 0.00 :        a44a:       je     a234 <_dl_lookup_symbol_x+0xb64>
 0.00 :        a450:       cmp    0x8(%rdx),%rax
 0.00 :        a454:       je     a3b3 <_dl_lookup_symbol_x+0xce3>
 0.00 :        a45a:       xor    %ecx,%ecx
 0.00 :        a45c:       add    $0x1,%ecx
 0.00 :        a45f:       cmp    -0xd0(%rbp),%ecx
 0.00 :        a465:       je     a234 <_dl_lookup_symbol_x+0xb64>
 0.00 :        a46b:       add    $0x8,%rdx
 0.00 :        a46f:       cmp    0x8(%rdx),%rax
 0.00 :        a473:       jne    a45c <_dl_lookup_symbol_x+0xd8c>
 0.00 :        a475:       jmpq   a3b3 <_dl_lookup_symbol_x+0xce3>
 0.00 :        a47a:       mov    -0xa0(%rbp),%rax
 0.00 :        a481:       xor    %edx,%edx
 0.00 :        a483:       jmpq   9ebb <_dl_lookup_symbol_x+0x7eb>
 0.00 :        a488:       testb  $0x8,0x3d4(%r15)
 0.00 :        a490:       jne    a416 <_dl_lookup_symbol_x+0xd46>
 0.00 :        a492:       mov    0x3c8(%r15),%eax
 0.00 :        a499:       cmp    -0xd0(%rbp),%eax
 0.00 :        a49f:       jbe    a52a <_dl_lookup_symbol_x+0xe5a>
 0.00 :        a4a5:       mov    -0xd0(%rbp),%edx
 0.00 :        a4ab:       mov    0x3c0(%r15),%rax
 0.00 :        a4b2:       mov    %r14,0x8(%rax,%rdx,8)
 0.00 :        a4b7:       mov    -0xd0(%rbp),%r8d
 0.00 :        a4be:       mov    0x3c0(%r15),%rax
 0.00 :        a4c5:       add    $0x1,%r8d
 0.00 :        a4c9:       mov    %r8d,(%rax)
 0.00 :        a4cc:       testb  $0x40,0x2176cd(%rip)        # 221ba0 <_rtld_global_ro>
 0.00 :        a4d3:       je     a420 <_dl_lookup_symbol_x+0xd50>
 0.00 :        a4d9:       mov    0x8(%r15),%rcx
 0.00 :        a4dd:       mov    0x30(%r15),%r8
 0.00 :        a4e1:       cmpb   $0x0,(%rcx)
 0.00 :        a4e4:       jne    a4f0 <_dl_lookup_symbol_x+0xe20>
 0.00 :        a4e6:       mov    0x2177eb(%rip),%rax        # 221cd8 <_dl_argv>
 0.00 :        a4ed:       mov    (%rax),%rcx
 0.00 :        a4f0:       mov    0x8(%r14),%rsi
 0.00 :        a4f4:       mov    0x30(%r14),%rdx
 0.00 :        a4f8:       cmpb   $0x0,(%rsi)
 0.00 :        a4fb:       jne    a507 <_dl_lookup_symbol_x+0xe37>
 0.00 :        a4fd:       mov    0x2177d4(%rip),%rax        # 221cd8 <_dl_argv>
 0.00 :        a504:       mov    (%rax),%rsi
 0.00 :        a507:       lea    0x12d02(%rip),%rdi        # 1d210 <__PRETTY_FUNCTION__.4816+0x16a7>
 0.00 :        a50e:       xor    %eax,%eax
 0.00 :        a510:       mov    %r9d,-0x110(%rbp)
 0.00 :        a517:       callq  f8f0 <_dl_debug_printf>
 0.00 :        a51c:       xor    %edx,%edx
 0.00 :        a51e:       mov    -0x110(%rbp),%r9d
 0.00 :        a525:       jmpq   a281 <_dl_lookup_symbol_x+0xbb1>
 0.00 :        a52a:       test   %eax,%eax
```

```
 0.00 :        a52c:       je     a654 <_dl_lookup_symbol_x+0xf84>
 0.00 :        a532:       add    %eax,%eax
 0.00 :        a534:       lea    0x8(,%rax,8),%rdi
 0.00 :        a53c:       mov    %eax,-0xe0(%rbp)
 0.00 :        a542:       mov    %fs:0x4c,%eax
 0.00 :        a54a:       test   %eax,%eax
 0.00 :        a54c:       je     a57b <_dl_lookup_symbol_x+0xeab>
 0.00 :        a54e:       mov    %rdi,-0x100(%rbp)
 0.00 :        a555:       mov    %r9d,-0x110(%rbp)
 0.00 :        a55c:       callq  14790 <_dl_x86_64_save_sse>
 0.00 :        a561:       movl   $0x0,%fs:0x4c
 0.00 :        a56d:       mov    -0x110(%rbp),%r9d
 0.00 :        a574:       mov    -0x100(%rbp),%rdi
 0.00 :        a57b:       mov    %r9d,-0x110(%rbp)
 0.00 :        a582:       callq  b50 <malloc@plt>
 0.00 :        a587:       test   %rax,%rax
 0.00 :        a58a:       mov    %rax,%rcx
 0.00 :        a58d:       mov    -0x110(%rbp),%r9d
 0.00 :        a594:       je     a645 <_dl_lookup_symbol_x+0xf75>
 0.00 :        a59a:       mov    -0xd0(%rbp),%r10d
 0.00 :        a5a1:       xor    %r8d,%r8d
 0.00 :        a5a4:       test   %r10d,%r10d
 0.00 :        a5a7:       je     a5f6 <_dl_lookup_symbol_x+0xf26>
 0.00 :        a5a9:       mov    -0xd0(%rbp),%r8d
 0.00 :        a5b0:       mov    0x3c0(%r15),%rsi
 0.00 :        a5b7:       lea    0x8(%rax),%rdi
 0.00 :        a5bb:       mov    %r9d,-0x110(%rbp)
 0.00 :        a5c2:       mov    %rax,-0x100(%rbp)
 0.00 :        a5c9:       add    $0x8,%rsi
 0.00 :        a5cd:       lea    0x0(,%r8,8),%rdx
 0.00 :        a5d5:       mov    %r8,-0x108(%rbp)
 0.00 :        a5dc:       callq  18a60 <memcpy>
 0.00 :        a5e1:       mov    -0x110(%rbp),%r9d
 0.00 :        a5e8:       mov    -0x108(%rbp),%r8
 0.00 :        a5ef:       mov    -0x100(%rbp),%rcx
 0.00 :        a5f6:       mov    %r14,0x8(%rcx,%r8,8)
 0.00 :        a5fb:       mov    -0xd0(%rbp),%r8d
 0.00 :        a602:       add    $0x1,%r8d
 0.00 :        a606:       mov    %r8d,(%rcx)
 0.00 :        a609:       mov    0x3c0(%r15),%rdi
 0.00 :        a610:       mov    %rcx,0x3c0(%r15)
 0.00 :        a617:       mov    -0xe0(%rbp),%ecx
 0.00 :        a61d:       test   %rdi,%rdi
 0.00 :        a620:       mov    %ecx,0x3c8(%r15)
 0.00 :        a627:       je     a4cc <_dl_lookup_symbol_x+0xdfc>
 0.00 :        a62d:       mov    %r9d,-0x110(%rbp)
 0.00 :        a634:       callq  11d60 <_dl_scope_free>
 0.00 :        a639:       mov    -0x110(%rbp),%r9d
 0.00 :        a640:       jmpq   a4cc <_dl_lookup_symbol_x+0xdfc>
 0.00 :        a645:       orl    $0x8,0x3d4(%r14)
 0.00 :        a64d:       xor    %edx,%edx
 0.00 :        a64f:       jmpq   a281 <_dl_lookup_symbol_x+0xbb1>
 0.00 :        a654:       mov    $0x58,%edi
 0.00 :        a659:       movl   $0xa,-0xe0(%rbp)
 0.00 :        a663:       jmpq   a542 <_dl_lookup_symbol_x+0xe72>
 0.00 :        a668:       mov    (%r8),%edx
 0.00 :        a66b:       cmp    -0xd0(%rbp),%edx
 0.00 :        a671:       jbe    a234 <_dl_lookup_symbol_x+0xb64>
 0.00 :        a677:       mov    -0xd0(%rbp),%ecx
 0.00 :        a67d:       cmp    0x8(%r8,%rcx,8),%rax
 0.00 :        a682:       je     a3b3 <_dl_lookup_symbol_x+0xce3>
 0.00 :        a688:       mov    -0xd0(%rbp),%ecx
 0.00 :        a68e:       add    $0x1,%ecx
 0.00 :        a691:       cmp    %ecx,%edx
```

```
    0.00 :          a693:       jbe    a6a3 <_dl_lookup_symbol_x+0xfd3>
    0.00 :          a695:       mov    %ecx,%esi
    0.00 :          a697:       cmp    0x8(%r8,%rsi,8),%rax
    0.00 :          a69c:       jne    a68e <_dl_lookup_symbol_x+0xfbe>
    0.00 :          a69e:       jmpq   a3b3 <_dl_lookup_symbol_x+0xce3>
    0.00 :          a6a3:       mov    %edx,-0xd0(%rbp)
    0.00 :          a6a9:       jmpq   a234 <_dl_lookup_symbol_x+0xb64>
 Percent |       Source code & Disassembly of ld-2.17.so
-----------------------------------------------
         :
         :
         :
         :       Disassembly of section .text:
         :
         :       000000000000c300 <_dl_map_object_deps>:
    0.00 :          c300:       push   %rbp
    0.00 :          c301:       lea    0x2(%rdx),%eax
    0.00 :          c304:       mov    %rsp,%rbp
    0.00 :          c307:       push   %r15
    0.00 :          c309:       lea    (%rax,%rax,2),%rax
    0.00 :          c30d:       push   %r14
    0.00 :          c30f:       lea    0x1e(,%rax,8),%rax
    0.00 :          c317:       push   %r13
    0.00 :          c319:       shr    $0x4,%rax
    0.00 :          c31d:       shl    $0x4,%rax
    0.00 :          c321:       push   %r12
    0.00 :          c323:       push   %rbx
    0.00 :          c324:       sub    $0xe8,%rsp
    0.00 :          c32b:       mov    %ecx,-0xb4(%rbp)
    0.00 :          c331:       mov    %rdi,-0xb0(%rbp)
    0.00 :          c338:       sub    %rax,%rsp
    0.00 :          c33b:       mov    %r8d,-0xb8(%rbp)
    0.00 :          c342:       lea    0xf(%rsp),%rcx
    0.00 :          c347:       and    $0xfffffffffffffff0,%rcx
    0.00 :          c34b:       lea    0x18(%rcx),%rax
    0.00 :          c34f:       mov    %rcx,-0xe8(%rbp)
    0.00 :          c356:       movl   $0x0,(%rcx)
    0.00 :          c35c:       mov    %rdi,0x8(%rcx)
    0.00 :          c360:       mov    %rax,0x10(%rcx)
    0.00 :          c364:       movzbl 0x314(%rdi),%eax
    0.00 :          c36b:       and    $0xffffff9f,%eax
    0.00 :          c36e:       or     $0x20,%eax
    0.00 :          c371:       test   %edx,%edx
    0.00 :          c373:       mov    %al,0x314(%rdi)
    0.00 :          c379:       je     cf07 <_dl_map_object_deps+0xc07>
    0.00 :          c37f:       lea    0x1(%rdx),%r10d
    0.00 :          c383:       mov    %rsi,%r8
    0.00 :          c386:       movl   $0x1,-0x80(%rbp)
    0.00 :          c38d:       mov    $0x1,%esi
    0.00 :          c392:       mov    $0x1,%edi
    0.00 :          c397:       add    $0x1,%edi
    0.00 :          c39a:       lea    (%rsi,%rsi,2),%rax
    0.00 :          c39e:       mov    (%r8),%r9
    0.00 :          c3a1:       mov    %edi,%esi
    0.00 :          c3a3:       add    $0x8,%r8
    0.00 :          c3a7:       lea    (%rsi,%rsi,2),%r11
    0.00 :          c3ab:       lea    (%rcx,%rax,8),%rax
    0.00 :          c3af:       lea    (%rcx,%r11,8),%r11
    0.00 :          c3b3:       movl   $0x0,(%rax)
    0.00 :          c3b9:       mov    %r9,0x8(%rax)
    0.00 :          c3bd:       mov    %r11,0x10(%rax)
    0.00 :          c3c1:       movzbl 0x314(%r9),%eax
    0.00 :          c3c9:       and    $0xffffff9f,%eax
    0.00 :          c3cc:       or     $0x20,%eax
```

```
  0.00 :        c3cf:       cmp    %r10d,%edi
  0.00 :        c3d2:       mov    %al,0x314(%r9)
  0.00 :        c3d9:       jne    c397 <_dl_map_object_deps+0x97>
  0.00 :        c3db:       mov    %edx,%edx
  0.00 :        c3dd:       mov    0x8(%rcx),%r14
  0.00 :        c3e1:       mov    %edi,-0x80(%rbp)
  0.00 :        c3e4:       lea    (%rdx,%rdx,2),%rax
  0.00 :        c3e8:       shl    $0x3,%rax
  0.00 :        c3ec:       add    %rcx,%rax
  0.00 :        c3ef:       movq   $0x0,-0x68(%rbp)
  0.00 :        c3f7:       mov    %rcx,-0xa0(%rbp)
  0.00 :        c3fe:       mov    %rax,-0x88(%rbp)
  0.00 :        c405:       movq   $0x0,0x10(%rax)
  0.00 :        c40d:       mov    0x216d59(%rip),%eax        # 22316c <rtld_errno>
  0.00 :        c413:       movq   $0x0,-0xc8(%rbp)
  0.00 :        c41e:       movl   $0x0,0x216d44(%rip)        # 22316c <rtld_errno>
  0.00 :        c428:       movq   $0x0,-0xc0(%rbp)
  0.00 :        c433:       mov    %eax,-0xcc(%rbp)
  0.00 :        c439:       cmpq   $0x0,0x2b8(%r14)
  0.00 :        c441:       mov    -0xa0(%rbp),%rcx
  0.00 :        c448:       movq   $0x0,-0x98(%rbp)
  0.00 :        c453:       movl   $0x1,(%rcx)
  0.00 :        c459:       je     c67c <_dl_map_object_deps+0x37c>
  0.00 :        c45f:       cmpq   $0x0,0x48(%r14)
  0.00 :        c464:       je     c597 <_dl_map_object_deps+0x297>
  0.00 :        c46a:       mov    0x68(%r14),%rax
  0.00 :        c46e:       mov    0x10(%r14),%rbx
  0.00 :        c472:       mov    -0xb4(%rbp),%ecx
  0.00 :        c478:       mov    -0xb8(%rbp),%esi
  0.00 :        c47e:       mov    %r14,-0x58(%rbp)
  0.00 :        c482:       mov    0x8(%rax),%rax
  0.00 :        c486:       mov    %ecx,-0x50(%rbp)
  0.00 :        c489:       mov    %esi,-0x4c(%rbp)
  0.00 :        c48c:       mov    %rax,-0x78(%rbp)
  0.00 :        c490:       mov    %rax,-0x48(%rbp)
  0.00 :        c494:       mov    (%rbx),%rax
  0.00 :        c497:       test   %rax,%rax
  0.00 :        c49a:       je     c5b3 <_dl_map_object_deps+0x2b3>
  0.00 :        c4a0:       mov    -0xa0(%rbp),%rdi
  0.00 :        c4a7:       xor    %r12d,%r12d
  0.00 :        c4aa:       mov    -0x98(%rbp),%r13
  0.00 :        c4b1:       mov    %rdi,-0xa8(%rbp)
  0.00 :        c4b8:       jmp    c4e1 <_dl_map_object_deps+0x1e1>
  0.00 :        c4ba:       nopw   0x0(%rax,%rax,1)
  0.00 :        c4c0:       test   %r13,%r13
  0.00 :        c4c3:       je     c4d1 <_dl_map_object_deps+0x1d1>
  0.00 :        c4c5:       mov    %r12d,%edx
  0.00 :        c4c8:       add    $0x1,%r12d
  0.00 :        c4cc:       mov    %rax,0x0(%r13,%rdx,8)
100.00 :        c4d1:       add    $0x10,%rbx
  0.00 :        c4d5:       mov    (%rbx),%rax
  0.00 :        c4d8:       test   %rax,%rax
  0.00 :        c4db:       je     c5b6 <_dl_map_object_deps+0x2b6>
  0.00 :        c4e1:       cmp    $0x1,%rax
  0.00 :        c4e5:       jne    cc49 <_dl_map_object_deps+0x949>
  0.00 :        c4eb:       mov    -0x78(%rbp),%r15
  0.00 :        c4ef:       add    0x8(%rbx),%r15
  0.00 :        c4f3:       mov    $0x24,%esi
  0.00 :        c4f8:       mov    %r15,%rdi
  0.00 :        c4fb:       callq  17680 <index>
  0.00 :        c500:       test   %rax,%rax
  0.00 :        c503:       jne    cd70 <_dl_map_object_deps+0xa70>
  0.00 :        c509:       mov    %r15,%rax
  0.00 :        c50c:       lea    -0x58(%rbp),%r8
```

```
 0.00 :        c510:       lea    -0x257(%rip),%rcx        # c2c0 <openaux>
 0.00 :        c517:       lea    -0x69(%rbp),%rdx
 0.00 :        c51b:       lea    -0x68(%rbp),%rsi
 0.00 :        c51f:       lea    -0x60(%rbp),%rdi
 0.00 :        c523:       mov    %rax,-0x40(%rbp)
 0.00 :        c527:       callq  e7b0 <_dl_catch_error>
 0.00 :        c52c:       mov    -0x68(%rbp),%r15
 0.00 :        c530:       test   %r15,%r15
 0.00 :        c533:       jne    c710 <_dl_map_object_deps+0x410>
 0.00 :        c539:       mov    -0x38(%rbp),%rax
 0.00 :        c53d:       testb  $0x60,0x314(%rax)
 0.00 :        c544:       jne    c4c0 <_dl_map_object_deps+0x1c0>
 0.00 :        c54a:       sub    $0x30,%rsp
 0.00 :        c54e:       mov    -0x88(%rbp),%rcx
 0.00 :        c555:       addl   $0x1,-0x80(%rbp)
 0.00 :        c559:       lea    0xf(%rsp),%rdx
 0.00 :        c55e:       and    $0xfffffffffffffff0,%rdx
 0.00 :        c562:       movq   $0x0,0x10(%rdx)
 0.00 :        c56a:       mov    %rax,0x8(%rdx)
 0.00 :        c56e:       movl   $0x0,(%rdx)
 0.00 :        c574:       mov    %rdx,0x10(%rcx)
 0.00 :        c578:       movzbl 0x314(%rax),%ecx
 0.00 :        c57f:       and    $0xffffff9f,%ecx
 0.00 :        c582:       or     $0x20,%ecx
 0.00 :        c585:       mov    %cl,0x314(%rax)
 0.00 :        c58b:       mov    %rdx,-0x88(%rbp)
 0.00 :        c592:       jmpq   c4c0 <_dl_map_object_deps+0x1c0>
 0.00 :        c597:       cmpq   $0x0,0x1e0(%r14)
 0.00 :        c59f:       jne    c46a <_dl_map_object_deps+0x16a>
 0.00 :        c5a5:       cmpq   $0x0,0x1d0(%r14)
 0.00 :        c5ad:       jne    c46a <_dl_map_object_deps+0x16a>
 0.00 :        c5b3:       xor    %r12d,%r12d
 0.00 :        c5b6:       cmpq   $0x0,-0x98(%rbp)
 0.00 :        c5be:       je     c635 <_dl_map_object_deps+0x335>
 0.00 :        c5c0:       lea    0x1(%r12),%r13d
 0.00 :        c5c5:       mov    -0x98(%rbp),%rsi
 0.00 :        c5cc:       mov    %r12d,%eax
 0.00 :        c5cf:       lea    0x1(%r13,%r13,1),%edi
 0.00 :        c5d4:       movq   $0x0,(%rsi,%rax,8)
 0.00 :        c5dc:       shl    $0x3,%rdi
 0.00 :        c5e0:       callq  b50 <malloc@plt>
 0.00 :        c5e5:       test   %rax,%rax
 0.00 :        c5e8:       mov    %rax,%rbx
 0.00 :        c5eb:       je     cf1c <_dl_map_object_deps+0xc1c>
 0.00 :        c5f1:       mov    %r14,(%rax)
 0.00 :        c5f4:       mov    %r13d,%eax
 0.00 :        c5f7:       mov    -0x98(%rbp),%rsi
 0.00 :        c5fe:       lea    0x0(,%rax,8),%r13
 0.00 :        c606:       lea    0x8(%rbx),%rdi
 0.00 :        c60a:       mov    %r13,%rdx
 0.00 :        c60d:       callq  18a60 <memcpy>
 0.00 :        c612:       lea    0x2(%r12),%eax
 0.00 :        c617:       mov    %r13,%rdx
 0.00 :        c61a:       mov    %rbx,%rsi
 0.00 :        c61d:       lea    (%rbx,%rax,8),%rdi
 0.00 :        c621:       callq  18a60 <memcpy>
 0.00 :        c626:       orb    $0x1,0x316(%r14)
 0.00 :        c62e:       mov    %rbx,0x3b8(%r14)
 0.00 :        c635:       mov    -0xa0(%rbp),%rsi
 0.00 :        c63c:       mov    (%rsi),%eax
 0.00 :        c63e:       test   %eax,%eax
 0.00 :        c640:       je     c66c <_dl_map_object_deps+0x36c>
 0.00 :        c642:       mov    -0xa0(%rbp),%rax
 0.00 :        c649:       nopl   0x0(%rax)
```

```
 0.00 :        c650:       mov    0x10(%rax),%rax
 0.00 :        c654:       test   %rax,%rax
 0.00 :        c657:       je     d298 <_dl_map_object_deps+0xf98>
 0.00 :        c65d:       mov    (%rax),%r9d
 0.00 :        c660:       test   %r9d,%r9d
 0.00 :        c663:       jne    c650 <_dl_map_object_deps+0x350>
 0.00 :        c665:       mov    %rax,-0xa0(%rbp)
 0.00 :        c66c:       mov    -0xa0(%rbp),%rdx
 0.00 :        c673:       mov    0x8(%rdx),%r14
 0.00 :        c677:       jmpq   c439 <_dl_map_object_deps+0x139>
 0.00 :        c67c:       cmpq   $0x0,0x3b8(%r14)
 0.00 :        c684:       jne    c45f <_dl_map_object_deps+0x15f>
 0.00 :        c68a:       cmp    %r14,-0xb0(%rbp)
 0.00 :        c691:       je     c45f <_dl_map_object_deps+0x15f>
 0.00 :        c697:       movzwl 0x2b2(%r14),%eax
 0.00 :        c69f:       test   %ax,%ax
 0.00 :        c6a2:       je     c45f <_dl_map_object_deps+0x15f>
 0.00 :        c6a8:       shl    $0x3,%rax
 0.00 :        c6ac:       cmp    -0xc8(%rbp),%rax
 0.00 :        c6b3:       mov    -0xc0(%rbp),%rsi
 0.00 :        c6ba:       mov    %rsi,-0x98(%rbp)
 0.00 :        c6c1:       jbe    c45f <_dl_map_object_deps+0x15f>
 0.00 :        c6c7:       add    $0xf,%rax
 0.00 :        c6cb:       and    $0xfffffffffffffff0,%rax
 0.00 :        c6cf:       lea    0x1e(%rax),%rdx
 0.00 :        c6d3:       and    $0x1ffff0,%edx
 0.00 :        c6d9:       sub    %rdx,%rsp
 0.00 :        c6dc:       lea    0xf(%rsp),%rdx
 0.00 :        c6e1:       and    $0xfffffffffffffff0,%rdx
 0.00 :        c6e5:       lea    (%rdx,%rax,1),%rcx
 0.00 :        c6e9:       cmp    %rcx,%rsi
 0.00 :        c6ec:       je     d2a4 <_dl_map_object_deps+0xfa4>
 0.00 :        c6f2:       mov    %rdx,-0x98(%rbp)
 0.00 :        c6f9:       mov    %rax,-0xc8(%rbp)
 0.00 :        c700:       mov    %rdx,-0xc0(%rbp)
 0.00 :        c707:       jmpq   c45f <_dl_map_object_deps+0x15f>
 0.00 :        c70c:       nopl   0x0(%rax)
 0.00 :        c710:       mov    %r15,%rdi
 0.00 :        c713:       mov    %eax,-0x110(%rbp)
 0.00 :        c719:       callq  17810 <strlen>
 0.00 :        c71e:       lea    0x1(%rax),%rdx
 0.00 :        c722:       add    $0x1f,%rax
 0.00 :        c726:       mov    %r15,%rsi
 0.00 :        c729:       and    $0xfffffffffffffff0,%rax
 0.00 :        c72d:       sub    %rax,%rsp
 0.00 :        c730:       lea    0xf(%rsp),%rdi
 0.00 :        c735:       and    $0xfffffffffffffff0,%rdi
 0.00 :        c739:       callq  18a60 <memcpy>
 0.00 :        c73e:       mov    -0x60(%rbp),%r12
 0.00 :        c742:       mov    %rax,%rbx
 0.00 :        c745:       mov    %r12,%rdi
 0.00 :        c748:       callq  17810 <strlen>
 0.00 :        c74d:       lea    0x1(%rax),%rdx
 0.00 :        c751:       add    $0x1f,%rax
 0.00 :        c755:       mov    %r12,%rsi
 0.00 :        c758:       and    $0xfffffffffffffff0,%rax
 0.00 :        c75c:       sub    %rax,%rsp
 0.00 :        c75f:       lea    0xf(%rsp),%rdi
 0.00 :        c764:       and    $0xfffffffffffffff0,%rdi
 0.00 :        c768:       callq  18a60 <memcpy>
 0.00 :        c76d:       cmpb   $0x0,-0x69(%rbp)
 0.00 :        c771:       mov    %rax,-0x60(%rbp)
 0.00 :        c775:       mov    -0x110(%rbp),%ecx
 0.00 :        c77b:       jne    d280 <_dl_map_object_deps+0xf80>
```

```
 0.00 :        c781:       movl   $0xffffffff,-0x78(%rbp)
 0.00 :        c788:       test   %ecx,%ecx
 0.00 :        c78a:       cmove  -0x78(%rbp),%ecx
 0.00 :        c78e:       mov    %rbx,-0x68(%rbp)
 0.00 :        c792:       mov    %ecx,-0x78(%rbp)
 0.00 :        c795:       mov    0x2169d0(%rip),%r8d        # 22316c <rtld_errno>
 0.00 :        c79c:       test   %r8d,%r8d
 0.00 :        c79f:       jne    c7b7 <_dl_map_object_deps+0x4b7>
 0.00 :        c7a1:       mov    -0xcc(%rbp),%edi
 0.00 :        c7a7:       test   %edi,%edi
 0.00 :        c7a9:       je     c7b7 <_dl_map_object_deps+0x4b7>
 0.00 :        c7ab:       mov    -0xcc(%rbp),%eax
 0.00 :        c7b1:       mov    %eax,0x2169b5(%rip)        # 22316c <rtld_errno>
 0.00 :        c7b7:       mov    -0xb0(%rbp),%rdi
 0.00 :        c7be:       mov    0x3b8(%rdi),%rdi
 0.00 :        c7c5:       test   %rdi,%rdi
 0.00 :        c7c8:       mov    %rdi,-0x98(%rbp)
 0.00 :        c7cf:       je     c7f5 <_dl_map_object_deps+0x4f5>
 0.00 :        c7d1:       mov    -0xb0(%rbp),%rdi
 0.00 :        c7d8:       movzbl 0x314(%rdi),%eax
 0.00 :        c7df:       and    $0x3,%eax
 0.00 :        c7e2:       cmp    $0x2,%al
 0.00 :        c7e4:       je     ceda <_dl_map_object_deps+0xbda>
 0.00 :        c7ea:       movq   $0x0,-0x98(%rbp)
 0.00 :        c7f5:       mov    -0x80(%rbp),%eax
 0.00 :        c7f8:       lea    0x1(%rax,%rax,1),%edi
 0.00 :        c7fc:       shl    $0x3,%rdi
 0.00 :        c800:       callq  b50 <malloc@plt>
 0.00 :        c805:       test   %rax,%rax
 0.00 :        c808:       mov    %rax,%r15
 0.00 :        c80b:       je     d262 <_dl_map_object_deps+0xf62>
 0.00 :        c811:       mov    -0x80(%rbp),%eax
 0.00 :        c814:       mov    -0xb0(%rbp),%rdi
 0.00 :        c81b:       xor    %r14d,%r14d
 0.00 :        c81e:       mov    -0xe8(%rbp),%rdx
 0.00 :        c825:       mov    -0xb4(%rbp),%r8d
 0.00 :        c82c:       add    $0x1,%eax
 0.00 :        c82f:       mov    %rdi,%rcx
 0.00 :        c832:       lea    (%r15,%rax,8),%rax
 0.00 :        c836:       mov    %rax,0x2b8(%rdi)
 0.00 :        c83d:       mov    -0x80(%rbp),%eax
 0.00 :        c840:       mov    %eax,0x2c0(%rdi)
 0.00 :        c846:       test   %r8d,%r8d
 0.00 :        c849:       mov    0x8(%rdx),%rax
 0.00 :        c84d:       jne    d13b <_dl_map_object_deps+0xe3b>
 0.00 :        c853:       mov    0x2b8(%rcx),%rsi
 0.00 :        c85a:       mov    %r14d,%edi
 0.00 :        c85d:       add    $0x1,%r14d
 0.00 :        c861:       mov    %rax,(%rsi,%rdi,8)
 0.00 :        c865:       mov    0x8(%rdx),%rax
 0.00 :        c869:       andb   $0x9f,0x314(%rax)
 0.00 :        c870:       mov    0x10(%rdx),%rdx
 0.00 :        c874:       test   %rdx,%rdx
 0.00 :        c877:       jne    c846 <_dl_map_object_deps+0x546>
 0.00 :        c879:       testb  $0x8,0x215321(%rip)        # 221ba1 <_rtld_global_ro+0x1>
 0.00 :        c880:       jne    d0c5 <_dl_map_object_deps+0xdc5>
 0.00 :        c886:       mov    -0xb0(%rbp),%rdi
 0.00 :        c88d:       mov    0x2b8(%rdi),%rsi
 0.00 :        c894:       mov    (%rsi),%rcx
 0.00 :        c897:       cmp    %rdi,%rcx
 0.00 :        c89a:       mov    %rcx,-0x80(%rbp)
 0.00 :        c89e:       jne    d0a6 <_dl_map_object_deps+0xda6>
 0.00 :        c8a4:       mov    -0xb0(%rbp),%rdi
 0.00 :        c8ab:       mov    0x3c0(%rdi),%r13
```

```
 0.00 :        c8b2:       test   %r13,%r13
 0.00 :        c8b5:       je     d07a <_dl_map_object_deps+0xd7a>
 0.00 :        c8bb:       cmp    $0x1,%r14d
 0.00 :        c8bf:       jbe    c8ed <_dl_map_object_deps+0x5ed>
 0.00 :        c8c1:       lea    -0x2(%r14),%eax
 0.00 :        c8c5:       lea    0x8(%rsi),%rdx
 0.00 :        c8c9:       lea    0x10(%rsi,%rax,8),%rsi
 0.00 :        c8ce:       mov    (%rdx),%rcx
 0.00 :        c8d1:       add    $0x8,%rdx
 0.00 :        c8d5:       movzbl 0x314(%rcx),%eax
 0.00 :        c8dc:       and    $0xffffff9f,%eax
 0.00 :        c8df:       or     $0x20,%eax
 0.00 :        c8e2:       cmp    %rsi,%rdx
 0.00 :        c8e5:       mov    %al,0x314(%rcx)
 0.00 :        c8eb:       jne    c8ce <_dl_map_object_deps+0x5ce>
 0.00 :        c8ed:       lea    0x8(%r13),%rcx
 0.00 :        c8f1:       mov    %rcx,-0x88(%rbp)
 0.00 :        c8f8:       mov    0x0(%r13),%ecx
 0.00 :        c8fc:       test   %ecx,%ecx
 0.00 :        c8fe:       je     c9d5 <_dl_map_object_deps+0x6d5>
 0.00 :        c904:       xor    %r12d,%r12d
 0.00 :        c907:       mov    %r13,%rbx
 0.00 :        c90a:       jmp    c920 <_dl_map_object_deps+0x620>
 0.00 :        c90c:       mov    0x3c0(%rcx),%rdi
 0.00 :        c913:       add    $0x1,%r12d
 0.00 :        c917:       cmp    %r12d,(%rdi)
 0.00 :        c91a:       jbe    c9d5 <_dl_map_object_deps+0x6d5>
 0.00 :        c920:       mov    %r12d,%edx
 0.00 :        c923:       mov    -0x80(%rbp),%rcx
 0.00 :        c927:       mov    0x8(%rbx,%rdx,8),%rax
 0.00 :        c92c:       testb  $0x60,0x314(%rax)
 0.00 :        c933:       je     c90c <_dl_map_object_deps+0x60c>
 0.00 :        c935:       mov    0x3c8(%rcx),%eax
 0.00 :        c93b:       mov    %rdx,-0xf0(%rbp)
 0.00 :        c942:       lea    0x8(,%rax,8),%rdi
 0.00 :        c94a:       callq  b50 <malloc@plt>
 0.00 :        c94f:       test   %rax,%rax
 0.00 :        c952:       mov    %rax,%r13
 0.00 :        c955:       mov    -0xf0(%rbp),%rdx
 0.00 :        c95c:       je     cb36 <_dl_map_object_deps+0x836>
 0.00 :        c962:       mov    -0x88(%rbp),%rsi
 0.00 :        c969:       lea    0x8(%rax),%rdi
 0.00 :        c96d:       shl    $0x3,%rdx
 0.00 :        c971:       callq  18a60 <memcpy>
 0.00 :        c976:       mov    -0x80(%rbp),%rcx
 0.00 :        c97a:       lea    0x1(%r12),%r9d
 0.00 :        c97f:       mov    %r12d,%esi
 0.00 :        c982:       mov    0x3c0(%rcx),%rdi
 0.00 :        c989:       mov    (%rdi),%eax
 0.00 :        c98b:       cmp    %r9d,%eax
 0.00 :        c98e:       jbe    d4dd <_dl_map_object_deps+0x11dd>
 0.00 :        c994:       mov    %r9d,%edx
 0.00 :        c997:       mov    %edx,%ecx
 0.00 :        c999:       mov    0x8(%rbx,%rcx,8),%rcx
 0.00 :        c99e:       testb  $0x60,0x314(%rcx)
 0.00 :        c9a5:       jne    c9b2 <_dl_map_object_deps+0x6b2>
 0.00 :        c9a7:       mov    %esi,%r8d
 0.00 :        c9aa:       add    $0x1,%esi
 0.00 :        c9ad:       mov    %rcx,0x8(%r13,%r8,8)
 0.00 :        c9b2:       add    $0x1,%edx
 0.00 :        c9b5:       cmp    %eax,%edx
 0.00 :        c9b7:       jne    c997 <_dl_map_object_deps+0x697>
 0.00 :        c9b9:       lea    -0x1(%r9,%rdx,1),%eax
 0.00 :        c9be:       sub    %r12d,%eax
```

```
 0.00 :        c9c1:       mov    %eax,%r12d
 0.00 :        c9c4:       mov    %esi,0x0(%r13)
 0.00 :        c9c8:       add    $0x1,%r12d
 0.00 :        c9cc:       cmp    %r12d,(%rdi)
 0.00 :        c9cf:       ja     c920 <_dl_map_object_deps+0x620>
 0.00 :        c9d5:       cmp    $0x1,%r14d
 0.00 :        c9d9:       jbe    d4b7 <_dl_map_object_deps+0x11b7>
 0.00 :        c9df:       mov    -0x80(%rbp),%rcx
 0.00 :        c9e3:       lea    -0x2(%r14),%edx
 0.00 :        c9e7:       mov    0x2b8(%rcx),%rsi
 0.00 :        c9ee:       lea    0x8(%rsi),%rax
 0.00 :        c9f2:       lea    0x10(%rsi,%rdx,8),%rcx
 0.00 :        c9f7:       mov    (%rax),%rdx
 0.00 :        c9fa:       add    $0x8,%rax
 0.00 :        c9fe:       andb   $0x9f,0x314(%rdx)
 0.00 :        ca05:       cmp    %rcx,%rax
 0.00 :        ca08:       jne    c9f7 <_dl_map_object_deps+0x6f7>
 0.00 :        ca0a:       mov    %r14d,%ebx
 0.00 :        ca0d:       mov    %r15,%rdi
 0.00 :        ca10:       lea    0x0(,%rbx,8),%rcx
 0.00 :        ca18:       mov    %rcx,%rdx
 0.00 :        ca1b:       mov    %rcx,-0x88(%rbp)
 0.00 :        ca22:       callq  18a60 <memcpy>
 0.00 :        ca27:       lea    (%rbx,%rbx,1),%rdx
 0.00 :        ca2b:       mov    %rsp,-0xa8(%rbp)
 0.00 :        ca32:       xor    %esi,%esi
 0.00 :        ca34:       lea    0x10(%rdx),%rax
 0.00 :        ca38:       shr    $0x4,%rax
 0.00 :        ca3c:       shl    $0x4,%rax
 0.00 :        ca40:       sub    %rax,%rsp
 0.00 :        ca43:       mov    %rsp,%rdi
 0.00 :        ca46:       mov    %rsp,-0x108(%rbp)
 0.00 :        ca4d:       callq  182e0 <memset>
 0.00 :        ca52:       lea    -0x1(%r14),%ecx
 0.00 :        ca56:       mov    -0x108(%rbp),%r9
 0.00 :        ca5d:       mov    %r14d,-0x90(%rbp)
 0.00 :        ca64:       mov    $0x1,%r8d
 0.00 :        ca6a:       mov    $0x2,%r10d
 0.00 :        ca70:       mov    %ecx,-0xa0(%rbp)
 0.00 :        ca76:       mov    $0x1,%ecx
 0.00 :        ca7b:       mov    %ecx,%r14d
 0.00 :        ca7e:       addw   $0x1,(%r9,%r8,2)
 0.00 :        ca84:       cmp    -0xa0(%rbp),%r14d
 0.00 :        ca8b:       lea    (%r15,%r8,8),%rdi
 0.00 :        ca8f:       mov    (%rdi),%rbx
 0.00 :        ca92:       jae    cad9 <_dl_map_object_deps+0x7d9>
 0.00 :        ca94:       mov    -0xa0(%rbp),%esi
 0.00 :        ca9a:       nopw   0x0(%rax,%rax,1)
 0.00 :        caa0:       mov    %esi,%r12d
 0.00 :        caa3:       lea    (%r15,%r12,8),%r13
 0.00 :        caa7:       mov    0x0(%r13),%rax
 0.00 :        caab:       mov    0x3b8(%rax),%rax
 0.00 :        cab2:       test   %rax,%rax
 0.00 :        cab5:       jne    cac9 <_dl_map_object_deps+0x7c9>
 0.00 :        cab7:       jmp    cad1 <_dl_map_object_deps+0x7d1>
 0.00 :        cab9:       nopl   0x0(%rax)
 0.00 :        cac0:       add    $0x8,%rax
 0.00 :        cac4:       cmp    %rdx,%rbx
 0.00 :        cac7:       je     cb3f <_dl_map_object_deps+0x83f>
 0.00 :        cac9:       mov    (%rax),%rdx
 0.00 :        cacc:       test   %rdx,%rdx
 0.00 :        cacf:       jne    cac0 <_dl_map_object_deps+0x7c0>
 0.00 :        cad1:       sub    $0x1,%esi
 0.00 :        cad4:       cmp    %esi,%r14d
```

```
 0.00 :        cad7:       jb     caa0 <_dl_map_object_deps+0x7a0>
 0.00 :        cad9:       cmp    %r10d,-0x90(%rbp)
 0.00 :        cae0:       mov    %r10d,%r14d
 0.00 :        cae3:       je     cbbc <_dl_map_object_deps+0x8bc>
 0.00 :        cae9:       mov    -0x90(%rbp),%edx
 0.00 :        caef:       mov    %r10d,%r8d
 0.00 :        caf2:       xor    %esi,%esi
 0.00 :        caf4:       lea    (%r9,%r8,2),%rdi
 0.00 :        caf8:       mov    %r8,-0xf8(%rbp)
 0.00 :        caff:       mov    %r9,-0x108(%rbp)
 0.00 :        cb06:       mov    %r10d,-0x100(%rbp)
 0.00 :        cb0d:       sub    %r10d,%edx
 0.00 :        cb10:       add    %rdx,%rdx
 0.00 :        cb13:       callq  182e0 <memset>
 0.00 :        cb18:       mov    -0x100(%rbp),%r10d
 0.00 :        cb1f:       mov    -0xf8(%rbp),%r8
 0.00 :        cb26:       mov    -0x108(%rbp),%r9
 0.00 :        cb2d:       add    $0x1,%r10d
 0.00 :        cb31:       jmpq   ca7e <_dl_map_object_deps+0x77e>
 0.00 :        cb36:       mov    -0x80(%rbp),%rcx
 0.00 :        cb3a:       jmpq   c90c <_dl_map_object_deps+0x60c>
 0.00 :        cb3f:       mov    %esi,%ecx
 0.00 :        cb41:       mov    %r10d,%r11d
 0.00 :        cb44:       mov    %r8,-0xf8(%rbp)
 0.00 :        cb4b:       sub    %r14d,%ecx
 0.00 :        cb4e:       lea    (%r15,%r11,8),%rsi
 0.00 :        cb52:       mov    %r9,-0x108(%rbp)
 0.00 :        cb59:       lea    0x0(,%rcx,8),%rdx
 0.00 :        cb61:       mov    %rcx,-0x110(%rbp)
 0.00 :        cb68:       mov    %r10d,-0x100(%rbp)
 0.00 :        cb6f:       mov    %r11,-0xf0(%rbp)
 0.00 :        cb76:       callq  18150 <memmove>
 0.00 :        cb7b:       mov    -0x108(%rbp),%r9
 0.00 :        cb82:       mov    -0xf0(%rbp),%r11
 0.00 :        cb89:       mov    -0x90(%rbp),%eax
 0.00 :        cb8f:       mov    %rbx,0x0(%r13)
 0.00 :        cb93:       mov    -0x110(%rbp),%rcx
 0.00 :        cb9a:       mov    -0xf8(%rbp),%r8
 0.00 :        cba1:       movzwl (%r9,%r11,2),%edx
 0.00 :        cba6:       mov    -0x100(%rbp),%r10d
 0.00 :        cbad:       sub    %r14d,%eax
 0.00 :        cbb0:       cmp    %eax,%edx
 0.00 :        cbb2:       jbe    cc14 <_dl_map_object_deps+0x914>
 0.00 :        cbb4:       mov    %r10d,%r14d
 0.00 :        cbb7:       jmpq   cae9 <_dl_map_object_deps+0x7e9>
 0.00 :        cbbc:       mov    -0xa8(%rbp),%rsp
 0.00 :        cbc3:       mov    -0x88(%rbp),%rcx
 0.00 :        cbca:       movq   $0x0,(%r15,%rcx,1)
 0.00 :        cbd2:       mov    -0x80(%rbp),%rcx
 0.00 :        cbd6:       orb    $0x1,0x316(%rcx)
 0.00 :        cbdd:       mov    %r15,0x3b8(%rcx)
 0.00 :        cbe4:       cmpq   $0x0,-0x98(%rbp)
 0.00 :        cbec:       je     cbfa <_dl_map_object_deps+0x8fa>
 0.00 :        cbee:       mov    -0x98(%rbp),%rdi
 0.00 :        cbf5:       callq  11d60 <_dl_scope_free>
 0.00 :        cbfa:       mov    -0x78(%rbp),%edx
 0.00 :        cbfd:       test   %edx,%edx
 0.00 :        cbff:       jne    d4e5 <_dl_map_object_deps+0x11e5>
 0.00 :        cc05:       lea    -0x28(%rbp),%rsp
 0.00 :        cc09:       pop    %rbx
 0.00 :        cc0a:       pop    %r12
 0.00 :        cc0c:       pop    %r13
 0.00 :        cc0e:       pop    %r14
 0.00 :        cc10:       pop    %r15
```

```
 0.00 :        cc12:       pop    %rbp
 0.00 :        cc13:       retq
 0.00 :        cc14:       lea    (%r9,%r8,2),%rdi
 0.00 :        cc18:       lea    (%r9,%r11,2),%rsi
 0.00 :        cc1c:       lea    (%rcx,%rcx,1),%rdx
 0.00 :        cc20:       movzwl (%r9,%r8,2),%ebx
 0.00 :        cc25:       callq  18150 <memmove>
 0.00 :        cc2a:       mov    -0x108(%rbp),%r9
 0.00 :        cc31:       mov    -0xf8(%rbp),%r8
 0.00 :        cc38:       mov    -0x100(%rbp),%r10d
 0.00 :        cc3f:       mov    %bx,(%r9,%r12,2)
 0.00 :        cc44:       jmpq   ca7e <_dl_map_object_deps+0x77e>
 0.00 :        cc49:       and    $0xfffffffffffffffd,%rax
 0.00 :        cc4d:       cmp    $0x7ffffffd,%rax
 0.00 :        cc53:       jne    c4d1 <_dl_map_object_deps+0x1d1>
 0.00 :        cc59:       mov    -0x78(%rbp),%r15
 0.00 :        cc5d:       add    0x8(%rbx),%r15
 0.00 :        cc61:       mov    $0x24,%esi
 0.00 :        cc66:       mov    %r15,%rdi
 0.00 :        cc69:       callq  17680 <index>
 0.00 :        cc6e:       test   %rax,%rax
 0.00 :        cc71:       jne    d2be <_dl_map_object_deps+0xfbe>
 0.00 :        cc77:       mov    %r15,%rsi
 0.00 :        cc7a:       cmpq   $0x7ffffffd,(%rbx)
 0.00 :        cc81:       mov    %rsi,-0x40(%rbp)
 0.00 :        cc85:       je     d417 <_dl_map_object_deps+0x1117>
 0.00 :        cc8b:       testb  $0x1,0x214f0e(%rip)        # 221ba0 <_rtld_global_ro>
 0.00 :        cc92:       jne    d3f1 <_dl_map_object_deps+0x10f1>
 0.00 :        cc98:       lea    -0x58(%rbp),%r8
 0.00 :        cc9c:       lea    -0x9e3(%rip),%rcx        # c2c0 <openaux>
 0.00 :        cca3:       lea    -0x69(%rbp),%rdx
 0.00 :        cca7:       lea    -0x68(%rbp),%rsi
 0.00 :        ccab:       lea    -0x60(%rbp),%rdi
 0.00 :        ccaf:       callq  e7b0 <_dl_catch_error>
 0.00 :        ccb4:       mov    -0x68(%rbp),%r15
 0.00 :        ccb8:       mov    %eax,-0x90(%rbp)
 0.00 :        ccbe:       test   %r15,%r15
 0.00 :        ccc1:       jne    cf59 <_dl_map_object_deps+0xc59>
 0.00 :        ccc7:       mov    -0xa8(%rbp),%rcx
 0.00 :        ccce:       sub    $0x30,%rsp
 0.00 :        ccd2:       lea    0xf(%rsp),%rdx
 0.00 :        ccd7:       mov    (%rcx),%rax
 0.00 :        ccda:       and    $0xfffffffffffffff0,%rdx
 0.00 :        ccde:       test   %r13,%r13
 0.00 :        cce1:       mov    %rax,(%rdx)
 0.00 :        cce4:       mov    0x8(%rcx),%rax
 0.00 :        cce8:       mov    %rax,0x8(%rdx)
 0.00 :        ccec:       mov    0x10(%rcx),%rax
 0.00 :        ccf0:       mov    %rax,0x10(%rdx)
 0.00 :        ccf4:       mov    -0x38(%rbp),%rax
 0.00 :        ccf8:       movl   $0x0,(%rcx)
 0.00 :        ccfe:       mov    %rax,0x8(%rcx)
 0.00 :        cd02:       je     cd14 <_dl_map_object_deps+0xa14>
 0.00 :        cd04:       mov    %r12d,%ecx
 0.00 :        cd07:       add    $0x1,%r12d
 0.00 :        cd0b:       mov    %rax,0x0(%r13,%rcx,8)
 0.00 :        cd10:       mov    -0x38(%rbp),%rax
 0.00 :        cd14:       testb  $0x60,0x314(%rax)
 0.00 :        cd1b:       je     d45d <_dl_map_object_deps+0x115d>
 0.00 :        cd21:       mov    %rdx,%rcx
 0.00 :        cd24:       jmp    cd3d <_dl_map_object_deps+0xa3d>
 0.00 :        cd26:       nopw   %cs:0x0(%rax,%rax,1)
 0.00 :        cd30:       cmp    %rax,0x8(%rsi)
 0.00 :        cd34:       je     cfe4 <_dl_map_object_deps+0xce4>
```

```
 0.00 :        cd3a:       mov    %rsi,%rcx
 0.00 :        cd3d:       mov    0x10(%rcx),%rsi
 0.00 :        cd41:       test   %rsi,%rsi
 0.00 :        cd44:       jne    cd30 <_dl_map_object_deps+0xa30>
 0.00 :        cd46:       mov    (%rdx),%rax
 0.00 :        cd49:       mov    -0xa8(%rbp),%rcx
 0.00 :        cd50:       mov    %rax,(%rcx)
 0.00 :        cd53:       mov    0x8(%rdx),%rax
 0.00 :        cd57:       mov    %rax,0x8(%rcx)
 0.00 :        cd5b:       mov    0x10(%rdx),%rax
 0.00 :        cd5f:       mov    %rax,0x10(%rcx)
 0.00 :        cd63:       jmpq   c4d1 <_dl_map_object_deps+0x1d1>
 0.00 :        cd68:       nopl   0x0(%rax,%rax,1)
 0.00 :        cd70:       xor    %esi,%esi
 0.00 :        cd72:       mov    %rax,%rdi
 0.00 :        cd75:       callq  75a0 <_dl_dst_count>
 0.00 :        cd7a:       test   %rax,%rax
 0.00 :        cd7d:       mov    %rax,-0x90(%rbp)
 0.00 :        cd84:       je     c509 <_dl_map_object_deps+0x209>
 0.00 :        cd8a:       mov    0x215050(%rip),%eax        # 221de0 <__libc_enable_secure>
 0.00 :        cd90:       test   %eax,%eax
 0.00 :        cd92:       jne    cec7 <_dl_map_object_deps+0xbc7>
 0.00 :        cd98:       mov    %r15,%rdi
 0.00 :        cd9b:       callq  17810 <strlen>
 0.00 :        cda0:       mov    0x338(%r14),%rdi
 0.00 :        cda7:       mov    %rax,%rsi
 0.00 :        cdaa:       mov    %rax,%rcx
 0.00 :        cdad:       test   %rdi,%rdi
 0.00 :        cdb0:       je     ce60 <_dl_map_object_deps+0xb60>
 0.00 :        cdb6:       cmp    $0xffffffffffffffff,%rdi
 0.00 :        cdba:       je     cec0 <_dl_map_object_deps+0xbc0>
 0.00 :        cdc0:       mov    %rax,-0x110(%rbp)
 0.00 :        cdc7:       mov    %rax,-0xf0(%rbp)
 0.00 :        cdce:       callq  17810 <strlen>
 0.00 :        cdd3:       mov    -0x110(%rbp),%rcx
 0.00 :        cdda:       mov    -0xf0(%rbp),%rsi
 0.00 :        cde1:       cmpq   $0x3,0x214dc7(%rip)        # 221bb0 <_rtld_global_ro+0x10>
 0.00 :        cde9:       mov    $0x3,%edx
 0.00 :        cdee:       cmovae 0x214dba(%rip),%rdx        # 221bb0 <_rtld_global_ro+0x10>
 0.00 :        cdf6:       cmp    %rax,%rdx
 0.00 :        cdf9:       cmovb  %rax,%rdx
 0.00 :        cdfd:       cmp    $0x4,%rdx
 0.00 :        ce01:       jbe    ce13 <_dl_map_object_deps+0xb13>
 0.00 :        ce03:       sub    $0x4,%rdx
 0.00 :        ce07:       imul   -0x90(%rbp),%rdx
 0.00 :        ce0f:       lea    (%rsi,%rdx,1),%rcx
 0.00 :        ce13:       lea    0x1e(%rcx),%rax
 0.00 :        ce17:       mov    %r15,%rsi
 0.00 :        ce1a:       xor    %ecx,%ecx
 0.00 :        ce1c:       mov    %r14,%rdi
 0.00 :        ce1f:       and    $0xfffffffffffffff0,%rax
 0.00 :        ce23:       sub    %rax,%rsp
 0.00 :        ce26:       lea    0xf(%rsp),%rdx
 0.00 :        ce2b:       and    $0xfffffffffffffff0,%rdx
 0.00 :        ce2f:       callq  7650 <_dl_dst_substitute>
 0.00 :        ce34:       cmpb   $0x0,(%rax)
 0.00 :        ce37:       jne    c50c <_dl_map_object_deps+0x20c>
 0.00 :        ce3d:       testb  $0x1,0x214d5c(%rip)        # 221ba0 <_rtld_global_ro>
 0.00 :        ce44:       je     c4d1 <_dl_map_object_deps+0x1d1>
 0.00 :        ce4a:       lea    0x10697(%rip),%rdi        # 1d4e8 <__PRETTY_FUNCTION__.4816+0x197f>
 0.00 :        ce51:       mov    %r15,%rsi
 0.00 :        ce54:       xor    %eax,%eax
 0.00 :        ce56:       callq  f8f0 <_dl_debug_printf>
 0.00 :        ce5b:       jmpq   c4d1 <_dl_map_object_deps+0x1d1>
```

```
 0.00 :        ce60:       mov    0x8(%r14),%rax
 0.00 :        ce64:       cmpb   $0x0,(%rax)
 0.00 :        ce67:       je     ce79 <_dl_map_object_deps+0xb79>
 0.00 :        ce69:       lea    0x215b28(%rip),%rax        # 222998 <_rtld_global+0x998>
 0.00 :        ce70:       cmp    %rax,%r14
 0.00 :        ce73:       jne    cf3a <_dl_map_object_deps+0xc3a>
 0.00 :        ce79:       mov    %rcx,-0x110(%rbp)
 0.00 :        ce80:       mov    %rsi,-0xf0(%rbp)
 0.00 :        ce87:       callq  11bf0 <_dl_get_origin>
 0.00 :        ce8c:       lea    -0x1(%rax),%rdx
 0.00 :        ce90:       mov    %rax,%rdi
 0.00 :        ce93:       mov    %rax,0x338(%r14)
 0.00 :        ce9a:       xor    %eax,%eax
 0.00 :        ce9c:       mov    -0x110(%rbp),%rcx
 0.00 :        cea3:       mov    -0xf0(%rbp),%rsi
 0.00 :        ceaa:       cmp    $0xfffffffffffffffd,%rdx
 0.00 :        ceae:       ja     cde1 <_dl_map_object_deps+0xae1>
 0.00 :        ceb4:       jmpq   cdce <_dl_map_object_deps+0xace>
 0.00 :        ceb9:       nopl   0x0(%rax)
 0.00 :        cec0:       xor    %eax,%eax
 0.00 :        cec2:       jmpq   cde1 <_dl_map_object_deps+0xae1>
 0.00 :        cec7:       lea    0x105f2(%rip),%rcx        # 1d4c0 <__PRETTY_FUNCTION__.4816+0x1957>
 0.00 :        cece:       xor    %edx,%edx
 0.00 :        ced0:       mov    %r15,%rsi
 0.00 :        ced3:       xor    %edi,%edi
 0.00 :        ced5:       callq  e580 <_dl_signal_error>
 0.00 :        ceda:       cmpq   $0x0,0x2b8(%rdi)
 0.00 :        cee2:       je     c7f5 <_dl_map_object_deps+0x4f5>
 0.00 :        cee8:       lea    0x111a1(%rip),%rcx        # 1e090 <__PRETTY_FUNCTION__.8976>
 0.00 :        ceef:       lea    0xe523(%rip),%rsi        # 1b419 <__PRETTY_FUNCTION__.9566+0x40>
 0.00 :        cef6:       lea    0x106e3(%rip),%rdi        # 1d5e0 <__PRETTY_FUNCTION__.4816+0x1a77>
 0.00 :        cefd:       mov    $0x1fd,%edx
 0.00 :        cf02:       callq  15980 <__GI___assert_fail>
 0.00 :        cf07:       mov    -0xb0(%rbp),%r14
 0.00 :        cf0e:       xor    %eax,%eax
 0.00 :        cf10:       movl   $0x1,-0x80(%rbp)
 0.00 :        cf17:       jmpq   c3ec <_dl_map_object_deps+0xec>
 0.00 :        cf1c:       mov    -0xb0(%rbp),%rdi
 0.00 :        cf23:       lea    0x10696(%rip),%rcx        # 1d5c0 <__PRETTY_FUNCTION__.4816+0x1a57>
 0.00 :        cf2a:       xor    %edx,%edx
 0.00 :        cf2c:       mov    0x8(%rdi),%rsi
 0.00 :        cf30:       mov    $0xc,%edi
 0.00 :        cf35:       callq  e580 <_dl_signal_error>
 0.00 :        cf3a:       lea    0x1114f(%rip),%rcx        # 1e090 <__PRETTY_FUNCTION__.8976>
 0.00 :        cf41:       lea    0xe4d1(%rip),%rsi        # 1b419 <__PRETTY_FUNCTION__.9566+0x40>
 0.00 :        cf48:       lea    0xfdd1(%rip),%rdi        # 1cd20 <__PRETTY_FUNCTION__.4816+0x11b7>
 0.00 :        cf4f:       mov    $0xfb,%edx
 0.00 :        cf54:       callq  15980 <__GI___assert_fail>
 0.00 :        cf59:       mov    %r15,%rdi
 0.00 :        cf5c:       callq  17810 <strlen>
 0.00 :        cf61:       lea    0x1(%rax),%rdx
 0.00 :        cf65:       add    $0x1f,%rax
 0.00 :        cf69:       mov    %r15,%rsi
 0.00 :        cf6c:       and    $0xfffffffffffffff0,%rax
 0.00 :        cf70:       sub    %rax,%rsp
 0.00 :        cf73:       lea    0xf(%rsp),%rdi
 0.00 :        cf78:       and    $0xfffffffffffffff0,%rdi
 0.00 :        cf7c:       callq  18a60 <memcpy>
 0.00 :        cf81:       mov    -0x60(%rbp),%r12
 0.00 :        cf85:       mov    %rax,%rbx
 0.00 :        cf88:       mov    %r12,%rdi
 0.00 :        cf8b:       callq  17810 <strlen>
 0.00 :        cf90:       lea    0x1(%rax),%rdx
 0.00 :        cf94:       add    $0x1f,%rax
```

```
 0.00 :        cf98:       mov    %r12,%rsi
 0.00 :        cf9b:       and    $0xfffffffffffffff0,%rax
 0.00 :        cf9f:       sub    %rax,%rsp
 0.00 :        cfa2:       lea    0xf(%rsp),%rdi
 0.00 :        cfa7:       and    $0xfffffffffffffff0,%rdi
 0.00 :        cfab:       callq  18a60 <memcpy>
 0.00 :        cfb0:       cmpb   $0x0,-0x69(%rbp)
 0.00 :        cfb4:       mov    %rax,-0x60(%rbp)
 0.00 :        cfb8:       je     cfc2 <_dl_map_object_deps+0xcc2>
 0.00 :        cfba:       mov    %r15,%rdi
 0.00 :        cfbd:       callq  b90 <free@plt>
 0.00 :        cfc2:       mov    -0x90(%rbp),%r10d
 0.00 :        cfc9:       mov    $0xffffffff,%edi
 0.00 :        cfce:       mov    %rbx,-0x68(%rbp)
 0.00 :        cfd2:       test   %r10d,%r10d
 0.00 :        cfd5:       cmovne -0x90(%rbp),%edi
 0.00 :        cfdc:       mov    %edi,-0x78(%rbp)
 0.00 :        cfdf:       jmpq   c795 <_dl_map_object_deps+0x495>
 0.00 :        cfe4:       mov    -0xa8(%rbp),%rsi
 0.00 :        cfeb:       mov    -0x88(%rbp),%rdi
 0.00 :        cff2:       mov    %rdx,0x10(%rsi)
 0.00 :        cff6:       mov    0x10(%rcx),%rsi
 0.00 :        cffa:       cmp    -0x88(%rbp),%rsi
 0.00 :        d001:       mov    0x10(%rsi),%rsi
 0.00 :        d005:       cmove  %rcx,%rdi
 0.00 :        d009:       mov    %rsi,0x10(%rcx)
 0.00 :        d00d:       mov    0x20(%rax),%rcx
 0.00 :        d011:       mov    %rdi,-0x88(%rbp)
 0.00 :        d018:       test   %rcx,%rcx
 0.00 :        d01b:       je     d025 <_dl_map_object_deps+0xd25>
 0.00 :        d01d:       mov    0x18(%rax),%rsi
 0.00 :        d021:       mov    %rsi,0x18(%rcx)
 0.00 :        d025:       mov    0x18(%rax),%rsi
 0.00 :        d029:       test   %rsi,%rsi
 0.00 :        d02c:       je     d032 <_dl_map_object_deps+0xd32>
 0.00 :        d02e:       mov    %rcx,0x20(%rsi)
 0.00 :        d032:       mov    0x8(%rdx),%rcx
 0.00 :        d036:       mov    0x20(%rcx),%rsi
 0.00 :        d03a:       mov    %rsi,0x20(%rax)
 0.00 :        d03e:       mov    %rax,0x20(%rcx)
 0.00 :        d042:       mov    0x20(%rax),%rsi
 0.00 :        d046:       test   %rsi,%rsi
 0.00 :        d049:       je     d04f <_dl_map_object_deps+0xd4f>
 0.00 :        d04b:       mov    %rax,0x18(%rsi)
 0.00 :        d04f:       mov    -0x88(%rbp),%rdi
 0.00 :        d056:       cmp    %rdi,-0xa8(%rbp)
 0.00 :        d05d:       mov    %rcx,0x18(%rax)
 0.00 :        d061:       mov    %rdx,-0xa8(%rbp)
 0.00 :        d068:       jne    c4d1 <_dl_map_object_deps+0x1d1>
 0.00 :        d06e:       mov    %rdx,-0x88(%rbp)
 0.00 :        d075:       jmpq   c4d1 <_dl_map_object_deps+0x1d1>
 0.00 :        d07a:       mov    %r14d,%ebx
 0.00 :        d07d:       mov    %r15,%rdi
 0.00 :        d080:       lea    0x0(,%rbx,8),%rcx
 0.00 :        d088:       mov    %rcx,%rdx
 0.00 :        d08b:       mov    %rcx,-0x88(%rbp)
 0.00 :        d092:       callq  18a60 <memcpy>
 0.00 :        d097:       cmp    $0x1,%r14d
 0.00 :        d09b:       jbe    cbc3 <_dl_map_object_deps+0x8c3>
 0.00 :        d0a1:       jmpq   ca27 <_dl_map_object_deps+0x727>
 0.00 :        d0a6:       lea    0x10fe3(%rip),%rcx        # 1e090 <__PRETTY_FUNCTION__.8976>
 0.00 :        d0ad:       lea    0xe365(%rip),%rsi        # 1b419 <__PRETTY_FUNCTION__.9566+0x40>
 0.00 :        d0b4:       lea    0x105a5(%rip),%rdi        # 1d660 <__PRETTY_FUNCTION__.4816+0x1af7>
 0.00 :        d0bb:       mov    $0x24e,%edx
```

```
 0.00 :         d0c0:       callq  15980 <__GI___assert_fail>
 0.00 :         d0c5:       mov    0x214f34(%rip),%rbx        # 222000 <_rtld_global>
 0.00 :         d0cc:       cmp    -0xb0(%rbp),%rbx
 0.00 :         d0d3:       jne    c886 <_dl_map_object_deps+0x586>
 0.00 :         d0d9:       test   %r14d,%r14d
 0.00 :         d0dc:       je     c886 <_dl_map_object_deps+0x586>
 0.00 :         d0e2:       lea    -0x1(%r14),%eax
 0.00 :         d0e6:       mov    %r14d,-0x80(%rbp)
 0.00 :         d0ea:       xor    %r12d,%r12d
 0.00 :         d0ed:       mov    %r15,%r14
 0.00 :         d0f0:       lea    0x8(,%rax,8),%rax
 0.00 :         d0f8:       mov    %rax,-0x88(%rbp)
 0.00 :         d0ff:       mov    0x2b8(%rbx),%rax
 0.00 :         d106:       mov    (%rax,%r12,1),%r13
 0.00 :         d10a:       cmp    %r13,%rbx
 0.00 :         d10d:       je     d122 <_dl_map_object_deps+0xe22>
 0.00 :         d10f:       mov    0x388(%r13),%rax
 0.00 :         d116:       test   %rax,%rax
 0.00 :         d119:       je     d154 <_dl_map_object_deps+0xe54>
 0.00 :         d11b:       mov    0x8(%rax),%esi
 0.00 :         d11e:       test   %esi,%esi
 0.00 :         d120:       je     d154 <_dl_map_object_deps+0xe54>
 0.00 :         d122:       add    $0x8,%r12
 0.00 :         d126:       cmp    -0x88(%rbp),%r12
 0.00 :         d12d:       jne    d0ff <_dl_map_object_deps+0xdff>
 0.00 :         d12f:       mov    %r14,%r15
 0.00 :         d132:       mov    -0x80(%rbp),%r14d
 0.00 :         d136:       jmpq   c886 <_dl_map_object_deps+0x586>
 0.00 :         d13b:       testb  $0x2,0x315(%rax)
 0.00 :         d142:       je     c853 <_dl_map_object_deps+0x553>
 0.00 :         d148:       subl   $0x1,0x2c0(%rcx)
 0.00 :         d14f:       jmpq   c869 <_dl_map_object_deps+0x569>
 0.00 :         d154:       cmpq   $0x0,0x1e0(%r13)
 0.00 :         d15c:       jne    d24b <_dl_map_object_deps+0xf4b>
 0.00 :         d162:       cmpq   $0x0,0x1d0(%r13)
 0.00 :         d16a:       jne    d24b <_dl_map_object_deps+0xf4b>
 0.00 :         d170:       mov    %r13,%rsi
 0.00 :         d173:       mov    %r14,%rdi
 0.00 :         d176:       callq  c240 <_dl_build_local_scope>
 0.00 :         d17b:       cmp    -0x80(%rbp),%eax
 0.00 :         d17e:       mov    %eax,%r15d
 0.00 :         d181:       ja     d215 <_dl_map_object_deps+0xf15>
 0.00 :         d187:       test   %eax,%eax
 0.00 :         d189:       je     d1cc <_dl_map_object_deps+0xecc>
 0.00 :         d18b:       mov    (%r14),%rdx
 0.00 :         d18e:       lea    0x8(%r14),%rcx
 0.00 :         d192:       andb   $0x9f,0x314(%rdx)
 0.00 :         d199:       xor    %edx,%edx
 0.00 :         d19b:       add    $0x1,%edx
 0.00 :         d19e:       cmp    %edx,%r15d
 0.00 :         d1a1:       jbe    d1cc <_dl_map_object_deps+0xecc>
 0.00 :         d1a3:       mov    (%rcx),%rsi
 0.00 :         d1a6:       andb   $0x9f,0x314(%rsi)
 0.00 :         d1ad:       test   %edx,%edx
 0.00 :         d1af:       je     d1c6 <_dl_map_object_deps+0xec6>
 0.00 :         d1b1:       mov    (%rcx),%rsi
 0.00 :         d1b4:       cmpq   $0x0,0xc0(%rsi)
 0.00 :         d1bc:       je     d1c6 <_dl_map_object_deps+0xec6>
 0.00 :         d1be:       orb    $0x80,0x315(%r13)
 0.00 :         d1c6:       add    $0x8,%rcx
 0.00 :         d1ca:       jmp    d19b <_dl_map_object_deps+0xe9b>
 0.00 :         d1cc:       mov    %eax,%ecx
 0.00 :         d1ce:       lea    0x10(,%rcx,8),%rdi
 0.00 :         d1d6:       mov    %rcx,-0x110(%rbp)
```

```
 0.00 :         d1dd:       callq  b50 <malloc@plt>
 0.00 :         d1e2:       test   %rax,%rax
 0.00 :         d1e5:       mov    %rax,0x388(%r13)
 0.00 :         d1ec:       mov    -0x110(%rbp),%rcx
 0.00 :         d1f3:       je     d234 <_dl_map_object_deps+0xf34>
 0.00 :         d1f5:       lea    0x10(%rax),%rdi
 0.00 :         d1f9:       lea    0x0(,%rcx,8),%rdx
 0.00 :         d201:       mov    %r15d,0x8(%rax)
 0.00 :         d205:       mov    %r14,%rsi
 0.00 :         d208:       mov    %rdi,(%rax)
 0.00 :         d20b:       callq  18a60 <memcpy>
 0.00 :         d210:       jmpq   d122 <_dl_map_object_deps+0xe22>
 0.00 :         d215:       lea    0x10e74(%rip),%rcx        # 1e090 <__PRETTY_FUNCTION__.8976>
 0.00 :         d21c:       lea    0xe1f6(%rip),%rsi        # 1b419 <__PRETTY_FUNCTION__.9566+0x40>
 0.00 :         d223:       lea    0xe1f9(%rip),%rdi        # 1b423 <__PRETTY_FUNCTION__.9566+0x4a>
 0.00 :         d22a:       mov    $0x235,%edx
 0.00 :         d22f:       callq  15980 <__GI___assert_fail>
 0.00 :         d234:       mov    0x8(%rbx),%rsi
 0.00 :         d238:       lea    0x103c9(%rip),%rcx        # 1d608 <__PRETTY_FUNCTION__.4816+0x1a9f>
 0.00 :         d23f:       xor    %edx,%edx
 0.00 :         d241:       mov    $0xc,%edi
 0.00 :         d246:       callq  e580 <_dl_signal_error>
 0.00 :         d24b:       mov    0x8(%r13),%rsi
 0.00 :         d24f:       lea    0x103da(%rip),%rcx        # 1d630 <__PRETTY_FUNCTION__.4816+0x1ac7>
 0.00 :         d256:       xor    %edx,%edx
 0.00 :         d258:       mov    $0x16,%edi
 0.00 :         d25d:       callq  e580 <_dl_signal_error>
 0.00 :         d262:       mov    -0xb0(%rbp),%rdi
 0.00 :         d269:       lea    0x10398(%rip),%rcx        # 1d608 <__PRETTY_FUNCTION__.4816+0x1a9f>
 0.00 :         d270:       xor    %edx,%edx
 0.00 :         d272:       mov    0x8(%rdi),%rsi
 0.00 :         d276:       mov    $0xc,%edi
 0.00 :         d27b:       callq  e580 <_dl_signal_error>
 0.00 :         d280:       mov    %r15,%rdi
 0.00 :         d283:       callq  b90 <free@plt>
 0.00 :         d288:       mov    -0x110(%rbp),%ecx
 0.00 :         d28e:       jmpq   c781 <_dl_map_object_deps+0x481>
 0.00 :         d293:       nopl   0x0(%rax,%rax,1)
 0.00 :         d298:       movl   $0x0,-0x78(%rbp)
 0.00 :         d29f:       jmpq   c795 <_dl_map_object_deps+0x495>
 0.00 :         d2a4:       add    %rax,-0xc8(%rbp)
 0.00 :         d2ab:       mov    %rdx,-0xc0(%rbp)
 0.00 :         d2b2:       mov    %rdx,-0x98(%rbp)
 0.00 :         d2b9:       jmpq   c45f <_dl_map_object_deps+0x15f>
 0.00 :         d2be:       xor    %esi,%esi
 0.00 :         d2c0:       mov    %rax,%rdi
 0.00 :         d2c3:       callq  75a0 <_dl_dst_count>
 0.00 :         d2c8:       test   %rax,%rax
 0.00 :         d2cb:       mov    %rax,-0xd8(%rbp)
 0.00 :         d2d2:       je     cc77 <_dl_map_object_deps+0x977>
 0.00 :         d2d8:       mov    0x214b01(%rip),%r11d        # 221de0 <__libc_enable_secure>
 0.00 :         d2df:       test   %r11d,%r11d
 0.00 :         d2e2:       jne    cec7 <_dl_map_object_deps+0xbc7>
 0.00 :         d2e8:       mov    %r15,%rdi
 0.00 :         d2eb:       callq  17810 <strlen>
 0.00 :         d2f0:       mov    0x338(%r14),%rdi
 0.00 :         d2f7:       mov    %rax,-0xe0(%rbp)
 0.00 :         d2fe:       mov    %rax,-0x90(%rbp)
 0.00 :         d305:       test   %rdi,%rdi
 0.00 :         d308:       je     d3b4 <_dl_map_object_deps+0x10b4>
 0.00 :         d30e:       cmp    $0xffffffffffffffff,%rdi
 0.00 :         d312:       je     d3ad <_dl_map_object_deps+0x10ad>
 0.00 :         d318:       callq  17810 <strlen>
 0.00 :         d31d:       cmpq   $0x3,0x21488b(%rip)        # 221bb0 <_rtld_global_ro+0x10>
```

```
 0.00 :         d325:       mov    $0x3,%edx
 0.00 :         d32a:       cmovae 0x21487e(%rip),%rdx        # 221bb0 <_rtld_global_ro+0x10>
 0.00 :         d332:       cmp    %rax,%rdx
 0.00 :         d335:       cmovae %rdx,%rax
 0.00 :         d339:       cmp    $0x4,%rax
 0.00 :         d33d:       jbe    d359 <_dl_map_object_deps+0x1059>
 0.00 :         d33f:       sub    $0x4,%rax
 0.00 :         d343:       imul   -0xd8(%rbp),%rax
 0.00 :         d34b:       add    -0xe0(%rbp),%rax
 0.00 :         d352:       mov    %rax,-0x90(%rbp)
 0.00 :         d359:       mov    -0x90(%rbp),%rax
 0.00 :         d360:       xor    %ecx,%ecx
 0.00 :         d362:       mov    %r15,%rsi
 0.00 :         d365:       mov    %r14,%rdi
 0.00 :         d368:       add    $0x1e,%rax
 0.00 :         d36c:       and    $0xfffffffffffffff0,%rax
 0.00 :         d370:       sub    %rax,%rsp
 0.00 :         d373:       lea    0xf(%rsp),%rdx
 0.00 :         d378:       and    $0xfffffffffffffff0,%rdx
 0.00 :         d37c:       callq  7650 <_dl_dst_substitute>
 0.00 :         d381:       cmpb   $0x0,(%rax)
 0.00 :         d384:       mov    %rax,%rsi
 0.00 :         d387:       jne    cc7a <_dl_map_object_deps+0x97a>
 0.00 :         d38d:       cmpq   $0x7ffffffd,(%rbx)
 0.00 :         d394:       jne    ce3d <_dl_map_object_deps+0xb3d>
 0.00 :         d39a:       lea    0x10197(%rip),%rcx        # 1d538 <__PRETTY_FUNCTION__.4816+0x19cf>
 0.00 :         d3a1:       xor    %edx,%edx
 0.00 :         d3a3:       mov    %r15,%rsi
 0.00 :         d3a6:       xor    %edi,%edi
 0.00 :         d3a8:       callq  e580 <_dl_signal_error>
 0.00 :         d3ad:       xor    %eax,%eax
 0.00 :         d3af:       jmpq   d31d <_dl_map_object_deps+0x101d>
 0.00 :         d3b4:       mov    0x8(%r14),%rax
 0.00 :         d3b8:       cmpb   $0x0,(%rax)
 0.00 :         d3bb:       je     d3cd <_dl_map_object_deps+0x10cd>
 0.00 :         d3bd:       lea    0x2155d4(%rip),%rax        # 222998 <_rtld_global+0x998>
 0.00 :         d3c4:       cmp    %rax,%r14
 0.00 :         d3c7:       jne    d503 <_dl_map_object_deps+0x1203>
 0.00 :         d3cd:       callq  11bf0 <_dl_get_origin>
 0.00 :         d3d2:       lea    -0x1(%rax),%rdx
 0.00 :         d3d6:       mov    %rax,%rdi
 0.00 :         d3d9:       mov    %rax,0x338(%r14)
 0.00 :         d3e0:       xor    %eax,%eax
 0.00 :         d3e2:       cmp    $0xfffffffffffffffd,%rdx
 0.00 :         d3e6:       ja     d31d <_dl_map_object_deps+0x101d>
 0.00 :         d3ec:       jmpq   d318 <_dl_map_object_deps+0x1018>
 0.00 :         d3f1:       mov    0x8(%r14),%rdx
 0.00 :         d3f5:       cmpb   $0x0,(%rdx)
 0.00 :         d3f8:       jne    d404 <_dl_map_object_deps+0x1104>
 0.00 :         d3fa:       mov    0x2148d7(%rip),%rax        # 221cd8 <_dl_argv>
 0.00 :         d401:       mov    (%rax),%rdx
 0.00 :         d404:       lea    0x10185(%rip),%rdi        # 1d590 <__PRETTY_FUNCTION__.4816+0x1a27>
 0.00 :         d40b:       xor    %eax,%eax
 0.00 :         d40d:       callq  f8f0 <_dl_debug_printf>
 0.00 :         d412:       jmpq   cc98 <_dl_map_object_deps+0x998>
 0.00 :         d417:       testb  $0x1,0x214782(%rip)        # 221ba0 <_rtld_global_ro>
 0.00 :         d41e:       jne    d491 <_dl_map_object_deps+0x1191>
 0.00 :         d420:       lea    -0x60(%rbp),%rdi
 0.00 :         d424:       lea    -0x58(%rbp),%r8
 0.00 :         d428:       lea    -0x116f(%rip),%rcx        # c2c0 <openaux>
 0.00 :         d42f:       lea    -0x69(%rbp),%rdx
 0.00 :         d433:       lea    -0x68(%rbp),%rsi
 0.00 :         d437:       callq  e7b0 <_dl_catch_error>
 0.00 :         d43c:       mov    -0x68(%rbp),%rdi
```

```
    0.00 :          d440:       test   %rdi,%rdi
    0.00 :          d443:       je     ccc7 <_dl_map_object_deps+0x9c7>
    0.00 :          d449:       cmpb   $0x0,-0x69(%rbp)
    0.00 :          d44d:       je     c4d1 <_dl_map_object_deps+0x1d1>
    0.00 :          d453:       callq  b90 <free@plt>
    0.00 :          d458:       jmpq   c4d1 <_dl_map_object_deps+0x1d1>
    0.00 :          d45d:       mov    -0xa8(%rbp),%rsi
    0.00 :          d464:       addl   $0x1,-0x80(%rbp)
    0.00 :          d468:       mov    %rdx,0x10(%rsi)
    0.00 :          d46c:       movzbl 0x314(%rax),%ecx
    0.00 :          d473:       and    $0xffffff9f,%ecx
    0.00 :          d476:       or     $0x20,%ecx
    0.00 :          d479:       mov    %cl,0x314(%rax)
    0.00 :          d47f:       mov    0x20(%rax),%rcx
    0.00 :          d483:       test   %rcx,%rcx
    0.00 :          d486:       jne    d01d <_dl_map_object_deps+0xd1d>
    0.00 :          d48c:       jmpq   d025 <_dl_map_object_deps+0xd25>
    0.00 :          d491:       mov    0x8(%r14),%rdx
    0.00 :          d495:       cmpb   $0x0,(%rdx)
    0.00 :          d498:       jne    d4a4 <_dl_map_object_deps+0x11a4>
    0.00 :          d49a:       mov    0x214837(%rip),%rax        # 221cd8 <_dl_argv>
    0.00 :          d4a1:       mov    (%rax),%rdx
    0.00 :          d4a4:       lea    0x100b5(%rip),%rdi        # 1d560 <__PRETTY_FUNCTION__.4816+0x19f7>
    0.00 :          d4ab:       xor    %eax,%eax
    0.00 :          d4ad:       callq  f8f0 <_dl_debug_printf>
    0.00 :          d4b2:       jmpq   d420 <_dl_map_object_deps+0x1120>
    0.00 :          d4b7:       mov    -0x80(%rbp),%rcx
    0.00 :          d4bb:       shl    $0x3,%r14
    0.00 :          d4bf:       mov    %r15,%rdi
    0.00 :          d4c2:       mov    %r14,%rdx
    0.00 :          d4c5:       mov    %r14,-0x88(%rbp)
    0.00 :          d4cc:       mov    0x2b8(%rcx),%rsi
    0.00 :          d4d3:       callq  18a60 <memcpy>
    0.00 :          d4d8:       jmpq   cbc3 <_dl_map_object_deps+0x8c3>
    0.00 :          d4dd:       mov    %r9d,%r12d
    0.00 :          d4e0:       jmpq   c9c4 <_dl_map_object_deps+0x6c4>
    0.00 :          d4e5:       cmpl   $0xffffffff,-0x78(%rbp)
    0.00 :          d4e9:       mov    $0x0,%eax
    0.00 :          d4ee:       mov    -0x68(%rbp),%rcx
    0.00 :          d4f2:       cmovne -0x78(%rbp),%eax
    0.00 :          d4f6:       mov    -0x60(%rbp),%rsi
    0.00 :          d4fa:       xor    %edx,%edx
    0.00 :          d4fc:       mov    %eax,%edi
    0.00 :          d4fe:       callq  e580 <_dl_signal_error>
    0.00 :          d503:       lea    0x10b86(%rip),%rcx        # 1e090 <__PRETTY_FUNCTION__.8976>
    0.00 :          d50a:       lea    0xdf08(%rip),%rsi        # 1b419 <__PRETTY_FUNCTION__.9566+0x40>
    0.00 :          d511:       lea    0xf808(%rip),%rdi        # 1cd20 <__PRETTY_FUNCTION__.4816+0x11b7>
    0.00 :          d518:       mov    $0x12f,%edx
    0.00 :          d51d:       callq  15980 <__GI___assert_fail>
 Percent |       Source code & Disassembly of ld-2.17.so
-----------------------------------------------
         :
         :
         :
         :       Disassembly of section .text:
         :
         :       00000000000170d0 <__GI___fxstat>:
    0.00 :          170d0:       cmp    $0x1,%edi
  100.00 :          170d3:       mov    %esi,%eax
    0.00 :          170d5:       ja     170f0 <__GI___fxstat+0x20>
    0.00 :          170d7:       movslq %eax,%rdi
    0.00 :          170da:       mov    %rdx,%rsi
    0.00 :          170dd:       mov    $0x5,%eax
    0.00 :          170e2:       syscall
```

```
    0.00 :          170e4:       cmp    $0xfffffffffffff000,%rax
    0.00 :          170ea:       ja     17100 <__GI___fxstat+0x30>
    0.00 :          170ec:       repz retq
    0.00 :          170ee:       xchg   %ax,%ax
    0.00 :          170f0:       movl   $0x16,0x20c072(%rip)        # 22316c <rtld_errno>
    0.00 :          170fa:       mov    $0xffffffff,%eax
    0.00 :          170ff:       retq
    0.00 :          17100:       neg    %eax
    0.00 :          17102:       mov    %eax,0x20c064(%rip)        # 22316c <rtld_errno>
    0.00 :          17108:       or     $0xffffffffffffffff,%rax
    0.00 :          1710c:       retq
 Percent |       Source code & Disassembly of [jbd2]
-----------------------------------------------
 Percent |       Source code & Disassembly of [jbd2]
-----------------------------------------------
 Percent |       Source code & Disassembly of [jbd2]
-----------------------------------------------
 Percent |       Source code & Disassembly of [jbd2]
-----------------------------------------------
 Percent |       Source code & Disassembly of [ext4]
-----------------------------------------------
 Percent |       Source code & Disassembly of [ext4]
-----------------------------------------------
 Percent |       Source code & Disassembly of [ext4]
-----------------------------------------------
 Percent |       Source code & Disassembly of [ext4]
-----------------------------------------------
 Percent |       Source code & Disassembly of [ext4]
-----------------------------------------------
```