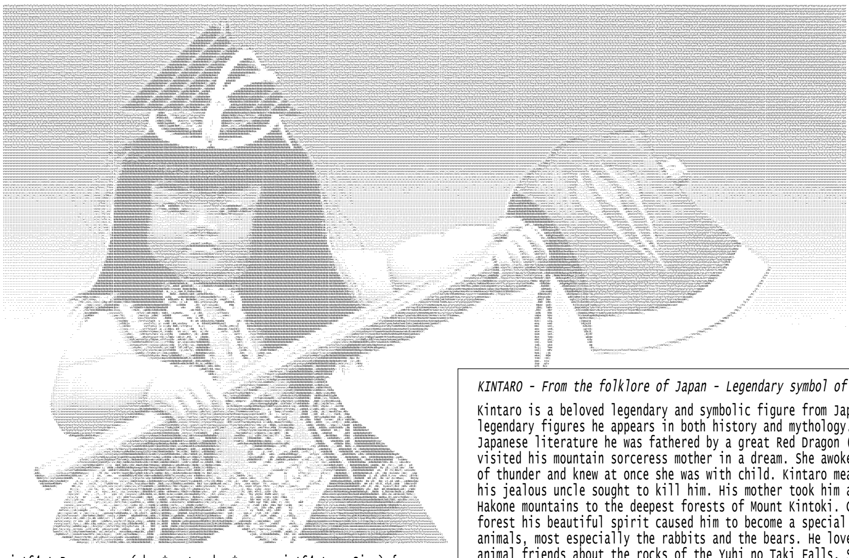
# 中道金太郎 – *NAKAMICHI 'Kintaro' a.k.a. 'Goldenboy'*

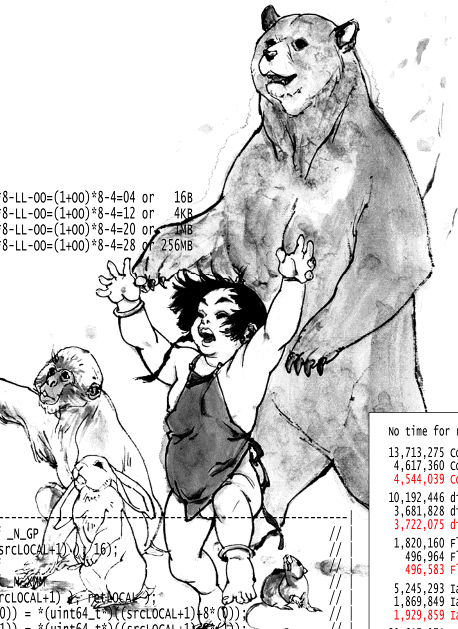
Fast&Strong for 30++MB textual files 100% FREE LZSS 64bit decompressor: *http://www.sanmayce.com/Hayabusa/index.html*
Old Home of Nakamichi: www.sanmayce.com/Nakamichi/index.html
Also: http://www.codeproject.com/Articles/878593/Slowest-LZSS-Compressor-in-C
Below, **Kintaro** (hero boy of Japanese folklore, who befriended animals and had supernatural strength) holding **Masakari**:

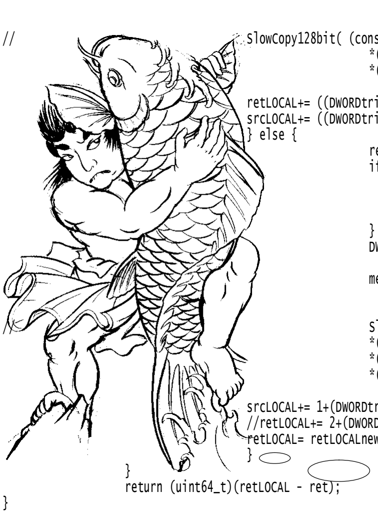


*KINTARO - From the folklore of Japan - Legendary symbol of virtue and strength*

Kintaro is a beloved legendary and symbolic figure from Japan. Like many legendary figures he appears in both history and mythology. According to classic Japanese literature he was fathered by a great Red Dragon (the thunder god) who visited his mountain sorceress mother in a dream. She awoke amidst powerful claps of thunder and knew at once she was with child. Kintaro means "Golden Boy" and his jealous uncle sought to kill him. His mother took him and fled into the Hakone mountains to the deepest forests of Mount Kintoki. Growing up deep in the forest his beautiful spirit caused him to become a special friend to all the wild animals, most especially the rabbits and the bears. He loved to play with his animal friends about the rocks of the Yuhi no Taki Falls. So strong was he as a boy and so gifted at Sumo wrestling that he could throw down a bear. He was a very good boy, rosy-cheeked and chubby and always carried a hatchet, the Japanese symbol of the thunder god and is usually depicted riding his beloved bear.

```
uint64_t Decompress (char* ret, char* src, uint64_t srcSize) {
                char* retLOCAL = ret; char* srcLOCAL = src;
                char* srcEndLOCAL = src+srcSize;
                unsigned int DWORDtrio; char* retLOCALnew;
                while (srcLOCAL < srcEndLOCAL) {
                                DWORDtrio = *(unsigned int*)srcLOCAL;
// |1stLSB     |2ndLSB  |3rdLSB   |
// -------------------------------
// |OO|LL|xxxx|xxxxxxxx|xxxxxx|xx|
// -------------------------------
// [1bit          16bit]    24bit]
// OOLL = 0000 means Literal - to ease the initial check (branchless also)
// LL = 00b means 00 MatchLength, (3+OO)+LL or 3+0 + 0|4|8|12 = 3| 7|11|15
// LL = 01b means 04 MatchLength, (3+OO)+LL or 3+1 + 0|4|8|12 = 4| 8|12|16
// LL = 10b means 08 MatchLength, (3+OO)+LL or 3+2 + 0|4|8|12 = 5| 9|13|17
// LL = 11b means 12 MatchLength, (3+OO)+LL or 3+3 + 0|4|8|12 = 6|10|14|18
// OO = 00b MatchOffset, 0xFFFFFFFF>>(3-OO), 1 bytes long i.e. Sliding Window is 1*8-LL-OO=(1+OO)*8-4=04 or   16B
// OO = 01b MatchOffset, 0xFFFFFFFF>>(3-OO), 2 bytes long i.e. Sliding Window is 2*8-LL-OO=(1+OO)*8-4=12 or   4KB
// OO = 10b MatchOffset, 0xFFFFFFFF>>(3-OO), 3 bytes long i.e. Sliding Window is 3*8-LL-OO=(1+OO)*8-4=20 or   1MB
// OO = 11b MatchOffset, 0xFFFFFFFF>>(3-OO), 4 bytes long i.e. Sliding Window is 4*8-LL-OO=(1+OO)*8-4=28 or 256MB
// (3+0 +  0):1 =  3:1 priority#11    #01  7:1 =  7 (OVERRIDE) with 24:2 = 12
// (3+0 +  4):1 =  7:1 priority#01    #02 16:2 =  8
// (3+0 +  8):1 = 11:1 priority#03    #03 11:1 = 11 (OVERRIDE) with 24:3 = 8
// (3+0 + 12):1 = 15:1 priority#05    #04 12:2 =  6
// (3+1 +  0):2 =  4:2 priority#14    #05 15:1 = 15 (OVERRIDE) with 24:4 = 6
// (3+1 +  4):2 =  8:2 priority#09    #06 17:3 =  5.6
// (3+1 +  8):2 = 12:2 priority#04    #07 18:4 =  4.5
// (3+1 + 12):2 = 16:2 priority#02    #08 13:3 =  4.3
// (3+2 +  0):3 =  5:3 priority#15    #09  8:2 =  4
// (3+2 +  4):3 =  9:3 priority#12    #10 14:4 =  3.5
// (3+2 +  8):3 = 13:3 priority#08    #11  3:1 =  3 (NOT USED)
// (3+2 + 12):3 = 17:3 priority#06    #12  9:3 =  3
// (3+3 +  0):4 =  6:4 priority#16    #13 10:4 =  2.5
// (3+3 +  4):4 = 10:4 priority#13    #14  4:2 =  2
// (3+3 +  8):4 = 14:4 priority#10    #15  5:3 =  1.6
// (3+3 + 12):4 = 18:4 priority#07    #16  6:4 =  1.5
                                if ( (DWORDtrio & 0x0F) == 0x00 ) { // ------------------------------------------------|
                                                                #ifdef _N_GP                                       // |
                                memcpy(retLOCAL, (const char *)( (uint64_t)(srcLOCAL+1) ), 16);                    // |
                                                                #endif                                             // |
                                                                #ifdef _N_XMM                                      // |
//                              SlowCopy128bit( (const char *)( (uint64_t)(srcLOCAL+1) ), retLOCAL );              // |
                                                *(uint64_t*)(retLOCAL+8*(0)) = *(uint64_t*)((srcLOCAL+1)+8*(0));   // |
                                                *(uint64_t*)(retLOCAL+8*(1)) = *(uint64_t*)((srcLOCAL+1)+8*(1));   // |
                                                                #endif                                             // |
                                retLOCAL+= ((DWORDtrio>>4)&0x0F);                                                  // |
                                srcLOCAL+= ((DWORDtrio>>4)&0x0F)+1;                                                // |
                                } else {                                                                           //\ /
                                                retLOCALnew = retLOCAL +3+(DWORDtrio&0x0F); // The idea is this second instance of
                                                if ( (DWORDtrio & 0x03) == 0x00 ) {
                                                                // 'DWORDtrio' has to be 'falsified' to look like the new offset
                                                                DWORDtrio = DWORDtrio|((DWORDtrio & 0x0F)>>2);
                                                                retLOCALnew = retLOCAL +24; // Total override, no mumbo-jumbo ar
                                                }
                                                DWORDtrio = DWORDtrio&( 0xFFFFFFFF >> ((3-(DWORDtrio & 0x03))<<3) );
                                                                #ifdef _N_GP
                                                memcpy(retLOCAL, (const char *)( (uint64_t)(retLOCAL-(DWORDtrio>>4)) ), 24);
                                                                #endif
                                                                #ifdef _N_XMM
//                                              SlowCopy128bit( (const char *)( (uint64_t)(retLOCAL-(DWORDtrio>>4)) ), retLOCAL );
                                                *(uint64_t*)(retLOCAL+8*(0)) = *(uint64_t*)((retLOCAL-(DWORDtrio>>4))+8*(0));
                                                *(uint64_t*)(retLOCAL+8*(1)) = *(uint64_t*)((retLOCAL-(DWORDtrio>>4))+8*(1));
                                                *(uint64_t*)(retLOCAL+8*(2)) = *(uint64_t*)((retLOCAL-(DWORDtrio>>4))+8*(2));
                                                                #endif
                                srcLOCAL+= 1+(DWORDtrio&0x03); // 4|3|2|1
                                //retLOCAL+= 2+(DWORDtrio&0x03) + (DWORDtrio&0x0C); // 2/3/4/5/6/7/8/9/10/11/12/13/14/15/16 // Hoshimi
                                retLOCAL= retLOCALnew; // 4/5/6, 8/9/10, 12/13/14, 16/17/18, 24 // Kintaro
                                }
                }
                return (uint64_t)(retLOCAL - ret);
}
```





Laptop Toshiba, Core 2 T7500 @2200MHz:

```
D:\Nakamichi_Kintaro>zstd -b dickens
dickens            :  10192446 ->   4070941 (39.94%),   83.5 MB/s ,  249.6 MB/s

D:\Nakamichi_Kintaro>lz4 -9 -b -T1 -Sx dickens
Nb of threads = 1 ; Compression Level = 9
dickens            : 10192446 ->   4442965 ( 43.59%),   10.2 MB/s ,  749.4 MB/s

D:\Nakamichi_Kintaro>Nakamichi_Kintaro_GCC_5.10.exe dickens.Nakamichi /report
Nakamichi 'Kintaro', written by Kaze, based on Nobuo Ito's LZSS source, babealicious suggestion by m^2 enforced, muffinesque suggestion by Jim Dempsey enforced.
Decompressing 3722075 bytes ...
RAM-to-RAM performance: 320 MB/s.
Compression Ratio (bigger-the-better): 2.74:1
```

Kintaro's firm grip of big textual blocks gives a new meaning of '**catching a big carp barehandedly**' (drawing by *Darko_Groenhagen*). **Nakamichi 'Kintaro'** is pure 64bit code and targets 256++MB English texts, that's it.

```
; mark_description "Intel(R) C++ Compiler XE for applications running on IA-32, Version 12.1.1.258 Build 20111011";
; 'Nakamichi_Kintaro_branchfull' decompression loop, 32bit:
; Size in bytes: b6-28+6=148
; Size in instructions: 53
```

```
; Goldenboy's main loop: b2-1e+6=154 bytes
; Goldenboy's main loop: 47 64bit instructions
; mark_description "Intel(R) C++ Intel(R) 64 Compiler XE for applications running on Intel(R) 64, Version 15.0.0.108 Build 20140";
; mark_description "-O3 -QxSSE2 -D_N_XMM -D_N_prefetch_4096 -D_N_Branchfull -FAcs";
```

## MASAKARI

鉞

*broadaxe*

The masakari is mainly used to smooth down and finish timber. There are two types of masakari. One has a longer handle and is used mainly for lumbering work, while the other type with a shorter handle, also called the carpenters' broadax (daiku-masakari), is used for rough carpentry work or for making wedges from scraps of wood.

A carpenter's hewing ax:

This ax was an essential tool for rough hewing work and for smoothing logs to be used as pillars and beams.
It is still used by carpenter's who work on shrines and temples.

Ono and Masakari as religious symbols:

The animistic tradition from ancient times state that deities descend to and reside in the mountains. For lumbermen, the mountain was therefore a sacred territory which required strict ritual abstentions to be entered. The ax has been closely related with this religious revering of the mountain and its trees. For example, the first act amongst the myriad of Shinto rituals carried out before the lumbering for the rebuilding of the Ise Shrine every 20 years, is the cutting into a tree with a ritually purified ax (imi-ono). Moreover in the festival of the pillar (Onbashira-matsuri) at the Suwa shrine, a vermillion-lacquered ax is used to cut down a tree which is to become the sacred pillar.

In Buddhist symbolism the ax also acquires the power of cutting off evil, and there are numerous existing statues of bodhisattva holding axes. Shugen-do, a traditional Japanese religion born out of an amalgam of different religions including Shintoism and Buddhism which has a particular connection with mountains, regards the ax as one of the symbolic objects to be carried by practitioners when going into mountains for ascetic training.

Ax is also an important (heavenly/carpentry) instrument in Laoism, a few excerpts from pseudo-chapter 74 of 'Dao De Jing':

Translation: *Lin Yutang*
And to take the place of the executioner
Is like handling the hatchet for the master carpenter.
He who handles the hatchet for the master carpenter
Seldom escapes injury to his hands.

Translation: *Gu Zhengkun*
If one wants to kill on behalf of the executioner,
It is like chopping wood on behalf of the master carpenter.
There are few who can escape cutting their own hands
When they chop wood on behalf of the master carpenter.

Translation: *Ch'u Ta-Kao*
Only the Supreme Executioner kills.
To kill in place of the Supreme Executioner is to hack instead of a greater carpenter.
Now if one hacks in place of a great carpenter one can scarcely

No time for richer picture, just the first dozen of files from '**Deathship**' corpus:

```
13,713,275 Complete_Works_of_Fyodor_Dostoyevsky.txt
 4,617,360 Complete_Works_of_Fyodor_Dostoyevsky.txt.gzip_mx9.zip
 4,544,039 Complete_Works_of_Fyodor_Dostoyevsky.txt.Nakamichi

10,192,446 dickens
 3,681,828 dickens.gzip_mx9.zip
 3,722,075 dickens.Nakamichi

 1,820,160 Fleurs_du_mal.tar
   496,964 Fleurs_du_mal.tar.gzip_mx9.zip
   496,583 Fleurs_du_mal.tar.Nakamichi

 5,245,293 Ian_Fleming_-_The_James_Bond_Anthology_(complete_collection).epub.txt
 1,869,849 Ian_Fleming_-_The_James_Bond_Anthology_(complete_collection).epub.txt.gzip_mx9.zip
 1,929,859 Ian_Fleming_-_The_James_Bond_Anthology_(complete_collection).epub.txt.Nakamichi

20,617,071 Large_traffic_log_file_of_a_popular_website_fp.log
 1,291,431 Large_traffic_log_file_of_a_popular_website_fp.log.gzip_mx9.zip
 2,243,699 Large_traffic_log_file_of_a_popular_website_fp.log.Nakamichi

    62,602 Roadhouse.sub
    22,874 Roadhouse.sub.gzip_mx9.zip
    35,477 Roadhouse.sub.Nakamichi

14,613,183 The_Book_of_The_Thousand_Nights_and_a_Night.txt
 5,198,949 The_Book_of_The_Thousand_Nights_and_a_Night.txt.gzip_mx9.zip
 5,228,912 The_Book_of_The_Thousand_Nights_and_a_Night.txt.Nakamichi

    92,096 The_Little_Prince_-_Antoine_de_Saint-Exupery.epub.txt
    30,329 The_Little_Prince_-_Antoine_de_Saint-Exupery.epub.txt.gzip_mx9.zip
    43,944 The_Little_Prince_-_Antoine_de_Saint-Exupery.epub.txt.Nakamichi

 7,137,280 The_Project_Gutenberg_12_Fairy_Books_by_Andrew_Lang.tar
 2,418,599 The_Project_Gutenberg_12_Fairy_Books_by_Andrew_Lang.tar.gzip_mx9.zip
 2,438,374 The_Project_Gutenberg_12_Fairy_Books_by_Andrew_Lang.tar.Nakamichi

 4,445,260 The_Project_Gutenberg_EBook_of_The_King_James_Bible_kjv10.txt
 1,320,100 The_Project_Gutenberg_EBook_of_The_King_James_Bible_kjv10.txt.gzip_mx9.zip
 1,420,630 The_Project_Gutenberg_EBook_of_The_King_James_Bible_kjv10.txt.Nakamichi

 3,265,536 University_of_Canterbury_The_Calgary_Corpus.tar
 1,017,658 University_of_Canterbury_The_Calgary_Corpus.tar.gzip_mx9.zip
 1,307,498 University_of_Canterbury_The_Calgary_Corpus.tar.Nakamichi

12,432,384 _Cambridge.History.of.Japan.6.Volumes.Set.PDF.ebooks.tar
 4,260,593 _Cambridge.History.of.Japan.6.Volumes.Set.PDF.ebooks.tar.gzip_mx9.zip
 4,361,794 _Cambridge.History.of.Japan.6.Volumes.Set.PDF.ebooks.tar.Nakamichi
```

*Igor Pavlov*'s **7z GZIP** was used. After 30+MB mark GZIP loses its edge, in 200+MB realm – DIES!

To kill instead of leaving killing to this power of death
is as if one wanted to use the axe oneself
instead of leaving it to the carpenter.
Whosoever would use the axe
instead of leaving it to the carpenter
shall rarely get away
without injuring his hand.

This documentlet is available thanks to next sources:

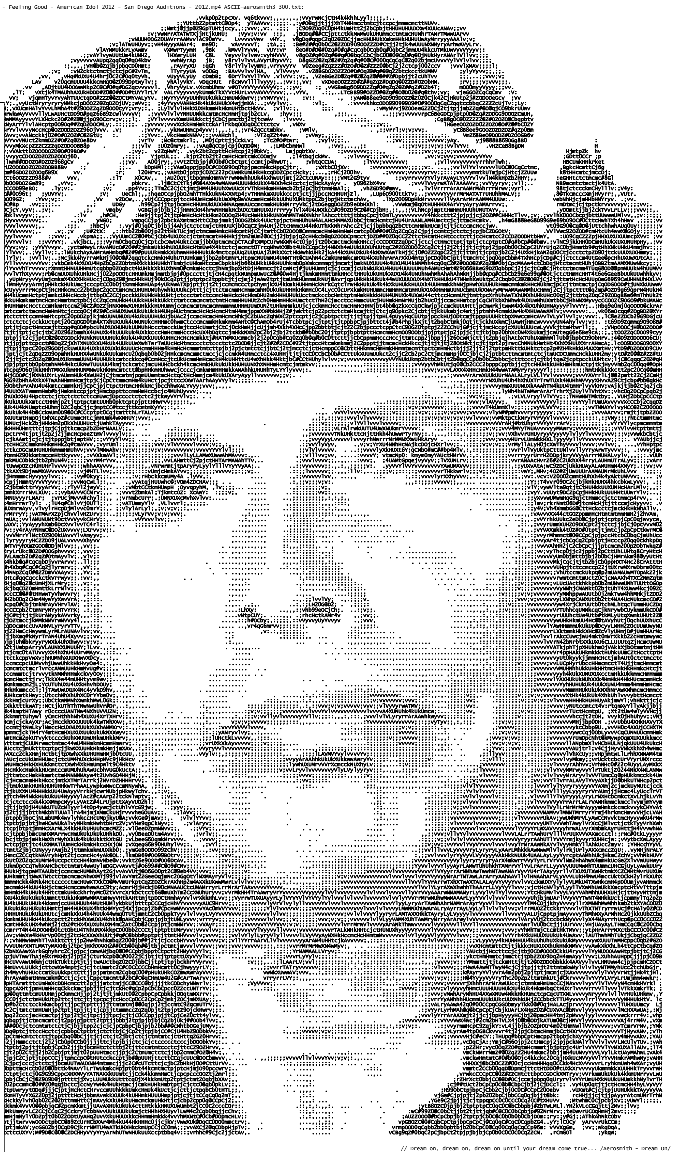
*1] TAKENAKA CARPENTRY TOOLS MUSEUM*
*4-18-25, Nakayamate-dori, Chuo-ku,*
*Kobe 650-0004, Japan*
*2] The Asian Myths & Legends Art page of Howard David Johnson*
*3] Japanese Architecture and Art Net Users System*

```
// D:\Nakamichi_Kintaro>gcc --version
// gcc (x86_64-posix-seh-rev0, Built by MinGW-W64 project) 5.1.0
// Copyright (C) 2015 Free Software Foundation, Inc.
// This is free software; see the source for copying conditions.  There is NO
// warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
//
// D:\Nakamichi_Kintaro>gcc -O3 -fomit-frame-pointer Nakamichi_Kintaro.c -o Nakamichi_Kintaro.exe -D_N_XMM -D_N_prefetch_4096 -D_N_Branchfull
/*
                .seh_proc          Decompress
Decompress:
                pushq              %rbp
                .seh_pushreg       %rbp
                pushq              %rdi
                .seh_pushreg       %rdi
                pushq              %rsi
                .seh_pushreg       %rsi
                pushq              %rbx
                .seh_pushreg       %rbx
                .seh_endprologue
                addq               %rdx, %r8
                movq               %rcx, %r11
                cmpq               %r8, %rdx
                jnb                .L23
                movq               %rcx, %rax
                movl               $-1, %esi
                movl               $8, %ebx
                movl               $16, %r10d
                jmp                .L22
                .p2align 4,,10
.L26:
                movq               1(%rdx), %rcx
                shrl               $4, %r9d
                andl               $15, %r9d
                movq               %rcx, (%rax)
                movq               9(%rdx), %rcx
                movq               %rcx, 8(%rax)
                movl               %r9d, %ecx
                addl               $1, %r9d
                addq               %r9, %rdx
                addq               %rcx, %rax
                cmpq               %r8, %rdx
                jnb                .L25
.L22:
                movl               (%rdx), %r9d
                movl               %r9d, %ecx
                andl               $15, %ecx
                je                 .L26
                testb              $3, %r9b
                je                 .L20
                movl               %ecx, %ecx
                leaq               3(%rax,%rcx), %rdi
.L21:
                movl               %r9d, %ecx
                movl               %esi, %ebp
                notl               %ecx
                andl               $3, %ecx
                sall               $3, %ecx
                shrl               %cl, %ebp
                andl               %ebp, %r9d
                movq               %rax, %rbp
                movl               %r9d, %ecx
                andl               $3, %r9d
                leaq               1(%rdx,%r9), %rdx
                shrl               $4, %ecx
                subq               %rcx, %rbp
                movq               0(%rbp), %rbp
                movq               %rbp, (%rax)
                movq               %rbx, %rbp
                subq               %rcx, %rbp
                movq               (%rax,%rbp), %rbp
                movq               %rbp, 8(%rax)
                movq               %r10, %rbp
                subq               %rcx, %rbp
                cmpq               %r8, %rdx
                movq               (%rax,%rbp), %rcx
                movq               %rcx, 16(%rax)
                movq               %rdi, %rax
                jb                 .L22
.L25:
                subq               %r11, %rax
.L17:
                popq               %rbx
                popq               %rsi
                popq               %rdi
                popq               %rbp
                ret
                .p2align 4,,10
.L20:
                leaq               24(%rax), %rdi
                shrl               $2, %ecx
                orl                %ecx, %r9d
                jmp                .L21
.L23:
                xorl               %eax, %eax
                jmp                .L17
                .seh_endproc
*/
```

```
// Goldenboy's main loop: b2-1e+6=154 bytes
// Goldenboy's main loop: 47 64bit instructions
/*
; mark_description "Intel(R) C++ Intel(R) 64 Compiler XE
for applications running on Intel(R) 64, Version 15.0.0.108
Build 20140";
; mark_description "-O3 -QxSSE2 -D_N_XMM -D_N_prefetch_4096
-D_N_Branchfull -FAcs";

       ALIGN     16
                    PUBLIC Decompress
Decompress          PROC
; parameter 1: rcx
; parameter 2: rdx
; parameter 3: r8
.B7.1::
  00000 48 83 ec 28      sub rsp, 40
  00004 49 89 d1         mov r9, rdx
  00007 4d 03 c1         add r8, r9
  0000a 49 89 ca         mov r10, rcx
  0000d 4c 89 d0         mov rax, r10
  00010 4d 3b c8         cmp r9, r8
  00013 0f 83 a4 00 00
        00               jae .B7.10 ; Prob 10%
.B7.2::
  00019 48 89 6c 24 20   mov QWORD PTR [32+rsp], rbp
.B7.3::
  0001e 41 8b 11         mov edx, DWORD PTR [r9]
  00021 41 89 d3         mov r11d, edx
  00024 41 83 e3 0f      and r11d, 15
  00028 75 26            jne .B7.5 ; Prob 50%
.B7.4::
  0002a c1 ea 04         shr edx, 4
  0002d 41 89 d3         mov r11d, edx
  00030 83 e2 0f         and edx, 15
  00033 ff c2            inc edx
  00035 49 83 e3 0f      and r11, 15
  00039 49 8b 49 01      mov rcx, QWORD PTR [1+r9]
  0003d 49 8b 69 09      mov rbp, QWORD PTR [9+r9]
  00041 48 89 08         mov QWORD PTR [rax], rcx
  00044 4c 03 ca         add r9, rdx
  00047 48 89 68 08      mov QWORD PTR [8+rax], rbp
  0004b 49 03 c3         add rax, r11
  0004e eb 5f            jmp .B7.8
.B7.5::
  00050 89 d1            mov ecx, edx
  00052 48 83 e1 0f      and rcx, 15
  00056 48 8d 6c 08 03   lea rbp, QWORD PTR [3+rax+rcx]
  0005b 89 d1            mov ecx, edx
  0005d 83 e1 03         and ecx, 3
  00060 75 0d            jne .B7.7 ; Prob 50%
.B7.6::
  00062 41 c1 eb 02      shr r11d, 2
  00066 48 8d 68 18      lea rbp, QWORD PTR [24+rax]
  0006a 41 0b d3         or edx, r11d
  0006d 89 d1            mov ecx, edx
.B7.7::
  0006f 83 f1 03         xor ecx, 3
  00072 41 bb ff ff ff
        ff               mov r11d, -1
  00078 c1 e1 03         shl ecx, 3
  0007b 41 d3 eb         shr r11d, cl
  0007e 41 23 d3         and edx, r11d
  00081 41 89 d3         mov r11d, edx
  00084 83 e2 03         and edx, 3
  00087 41 c1 eb 04      shr r11d, 4
  0008b ff c2            inc edx
  0008d 49 f7 db         neg r11
  00090 4c 03 d8         add r11, rax
  00093 4c 03 ca         add r9, rdx
  00096 49 8b 0b         mov rcx, QWORD PTR [r11]
  00099 48 89 08         mov QWORD PTR [rax], rcx
  0009c 49 8b 4b 08      mov rcx, QWORD PTR [8+r11]
  000a0 48 89 48 08      mov QWORD PTR [8+rax], rcx
  000a4 4d 8b 5b 10      mov r11, QWORD PTR [16+r11]
  000a8 4c 89 58 10      mov QWORD PTR [16+rax], r11
  000ac 48 89 e8         mov rax, rbp
.B7.8::
  000af 4d 3b c8         cmp r9, r8
  000b2 0f 82 66 ff ff
        ff               jb .B7.3 ; Prob 82%
.B7.9::
  000b8 48 8b 6c 24 20   mov rbp, QWORD PTR [32+rsp]
;
.B7.10::
  000bd 49 2b c2         sub rax, r10
  000c0 48 83 c4 28      add rsp, 40
  000c4 c3               ret
  000c5 0f 1f 40 00 0f
        1f 80 00 00 00
        00               ALIGN     16
.B7.11::
Decompress ENDP
```

// A still taken from Aubree Dieckmeyer - Feeling Good - American Idol 2012 - San Diego Auditions - 2012.mp4_ASCII-aerosmith3_300.txt:
/*



// Dream on, dream on, dream on until your dream come true... /Aerosmith - Dream On/