Project 'GAMERA'

(Semi-Powerful Console(Windows & Linux) Tools & gigabytes of English texts, downloadable from www.sanmayce.com)

WHERE THE WORD COUNTS

Caterpillar(LZSS-King of Brute-Force Heavy Sentence Dumpers, 32bit console application), revision 14+
Kazuya(LZ-Sovereign of Brute-Force Heavy Sentence Dumpers, 32bit console application), revision **17++**
Salah-ed-din(GZ-Sultan of Brute-Force Heavy Sentence Dumpers, 32bit console application), revision 14++
Raccoondog(LZMA-Baron of Brute-Force Heavy Sentence Dumpers, 32bit console application), revision **17++**
Yoshi(Filelist creator and more, 32bit console application), revision 06
Leprechaun(Fast and Greedy Word_Ripper, 32bit console application), revision **13++**

WinRAR archive in eleven 624MB volumes • Required HDD space: 6.56 GB (*ready to go when extracted on* **D:**\) • 2010 JUN 06

Kazuya delivers english sentences at 85-255MB/s speed(Obtained with Toshiba Satellite L305 (Intel Pentium(Merom-1M) T3400 2.16GHz))
Salah-ed-din delivers english sentences at 114-117MB/s speed(Obtained with Toshiba Satellite L305 (Intel Pentium(Merom-1M) T3400 2.16GHz))
Raccoondog delivers english sentences at 39MB/s speed(Obtained with Toshiba Satellite L305 (Intel Pentium(Merom-1M) T3400 2.16GHz))
Leprechaun rips 6,142,696++ words per second(Obtained with Toshiba Satellite L305 (Intel Pentium(Merom-1M) T3400 2.16GHz))
LBL stands for Line-By-Line(GRAMMATICAL ENGLISH LINES) i.e. sentences not merely CRLF or LF lines!
.LBL files are made from .TXT files which are made from respective .DOC, .RTF, .LIT, .PDF, .CHM, .HTM[L], .DJV[U] files;
Number and size of *.LBL files: 562,504 files(26GB or 27,991,747,152 bytes);
Lines and words in *.LBL files: 424,754,717 lines(with 4,582,451,898 words of them 9,177,221 distinct);

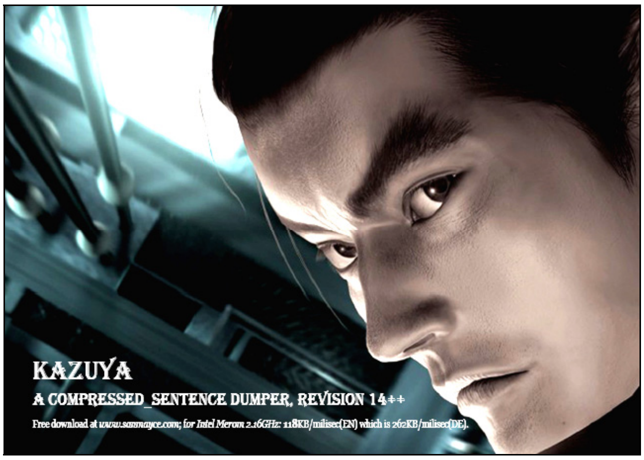
'**Monstrous Dumpers**' package, revision 13-

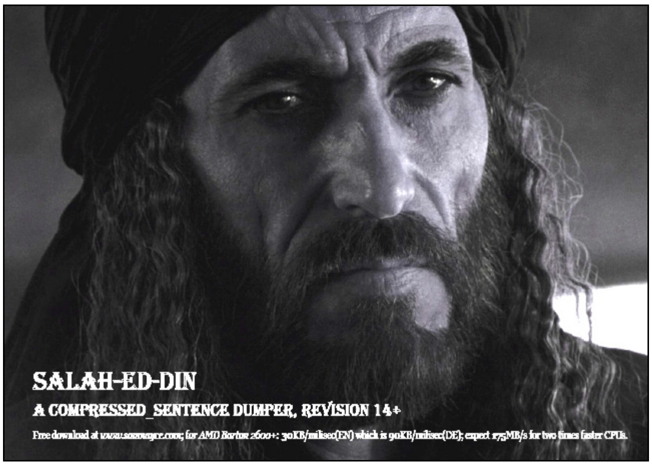
OR

<u>How words can be mixed into sentences!?</u>

With this package(the main part of project "GAMERA") you can make full-text(brute-force) requests into millions of lines(sentences). For example: make a search for **requests???????????into** to see whether that preposition has place near on right side of "requests".
This package(a Winrar archive) is intended as shareware and contains six very fast 32bit console text tools: **Caterpillar** (its rivals **Raccoondog**, **Salah-ed-din** and **Kazuya**), **Leprechaun**, **Yoshi** and of course 100++ million sentences(in English language) from various sources.

The package allows easily to create:
- a FILELIST(a text file with filenames);
- a WORDLIST(a text file with sorted distinct words);
- and as a main feature a text-pattern to be searched into LF(Unix)|CRLF(Windows) lines(or files) via filelist and to dump resultant hits(lines or filenames) into .HTML file.





Main features:

- 26GB english-ASCII-texts converted to .LBL(same as .TXT but each line is a sentence) format;
- File-by-file listings of all texts included:

```
    164,128 KAZE_G.S._Corpus_'.chm'_Caterpillar.html
     16,817 KAZE_G.S._Corpus_'.djv-'_Caterpillar.html
     61,142 KAZE_G.S._Corpus_'.doc'_Caterpillar.html
 22,742,832 KAZE_G.S._Corpus_'.htm-'_Caterpillar.html
     32,717 KAZE_G.S._Corpus_'.lit'_Caterpillar.html
  1,905,255 KAZE_G.S._Corpus_'.pdf'_Caterpillar.html
     30,535 KAZE_G.S._Corpus_'.rtf'_Caterpillar.html
 27,389,047 KAZE_G.S._Corpus_'.txt'_Caterpillar.html
```

- [Five(*,@,#,$,%):**Kazuya** & **Raccoondog**] Wildcards(*,?) available for patterns: very slow(pattern ***underdog*** took 490 seconds to look for into 400+ million sentences) but powerful;
- Fast(for laptops with a non-SSD disk)(90++MB/s) full-text traversing due to **zlib** used;
- Extra Fast(for laptops with a SSD disk)(200++MB/s) full-text traversing due to **QuickLZ** used;
- Results are delivered as *screen output* immediately and as *pure HTML files* finally;
- 'Karp_Rabin_Kaze'(patterns ***, underdog** took 70 seconds to look for into 400+ million sentences) compared to 'strstr'(85s) & 'Boyer-Moore-Horspool'(86s) is ((85-70)/70)*100%=21% faster when running on sentences.

Installation(i.e. extracting) notes:

- Unrar in D:\ if possible, "Caterpillar.lnk", "Go to PROMPT.lnk", "Raccoondog.lnk", "Salah-ed-din.lnk", "Kazuya.lnk" need manual adjustments if not D:\, 7GB must be free.
- To use "**Caterpillar.lnk**" and "**Salah-ed-din.lnk**" and "**Kazuya.lnk**" must run (R2O.BAT) and (R2G.BAT) and (R2L.BAT) respectively.

Current revisions of tools:

- ***EXEs(Windows):***

Caterpillar	r.14+
Leprechaun	r.13++
Yoshi	r.06
Salah-ed-din	r.14++
Raccoondog	r.17++
Kazuya	r.17++ r.15 has an ability to search non-compressed files too!

- ***ELFs(Linux):***

Caterpillar	r.14+
Leprechaun	r.13++
Yoshi	r.06
Salah-ed-din	r.14+

Note1: Revisions 14++ are Experimental(but operational, not beta) Karp-Rabin function with my hash, see last page.
Note2: Predecessor of *Caterpillar*, *Salah-ed-din* & *Raccoondog* was **Kazuya**(with more functionality and critical parts written in 16bit assembler), someday I will resurrect him in 64bit.

Convert at will:

Use G2R.BAT for .gz -> .lzma (1000+ minutes needed to convert, grmbl)
Use R2G.BAT for .lzma -> .gz (11:05 PM - 12:18 AM i.e 73 minutes needed to convert)
Use R2L.BAT for .lzma -> .Lasse (06:57 PM - 07:45 PM i.e 48 minutes needed to convert)
Use R2O.BAT for .lzma -> .Okumura (11:33 PM - 01:22 AM i.e 109 minutes needed to convert)

Some experience(Machine: Toshiba Satellite L305 - Intel Pentium Dual CPU T3400 @ 2.16GHz):

- **Caterpillar** uses **LZSS**(based on LZSS.C written by H.Okumura) compression;
  24.9GB -> 11.4 GB (12,341,932,922 bytes);
  delivering text at 82KB(*149KB when in system cache*)/clock i.e. 80MB/s;
  suitable for FAST HDDs 80+MB/s.
- **Raccoondog** uses **LZMA**(based on LZMA SDK 4.65 written by I.Pavlov) compression;
  24.9GB -> 5.5 GB (5,944,631,607 bytes);
  delivering text at 40KB(*bottleneck is CPU power alone*)/clock i.e. 39MB/s;
  suitable for flash cards like CFs, SDs.
- **Salah-ed-din** uses **GZ**(based on zlib 1.2.3 written by J.Gailly and M.Adler) compression;
  24.9GB -> 8.4 GB (9,051,049,655 bytes);
  delivering text at 117KB(*120KB when in system cache*)/clock i.e. 114MB/s;
  suitable for FAST CPUs 3+GHz.
- **Kazuya** uses **LZ**(based on QuickLZ 1.4.0 written by Lasse Reinhold) compression;
  24.9GB -> 10.6 GB (11,402,975,168 bytes);
  delivering text at 88KB(*262KB(118KB(EN)) when in system cache*)/clock i.e. 85MB/s;
  suitable for FAST SSDs 115+MB/s. Near future dreams: CPU(2x faster) and SSD(2x115Mb/s read) will give 2x255Mb/s.

Search|Seek|Find in order to Explore|Learn|Avoid Different Styles:

["Супруга съм на три деца. С чувекъ сбрахме пари и купихми триустаен партамент. Една вечер звъни вратата. Звънецъ чука. Отварям - НИНДЖА. И без да каже нищо, с карате в бъбреците. Дукат съ усета ми би два шамара с КРАК и един на детето в гръбначнийъ кош! От ударната вълна отльитам на 20-30 метра. Абстрахираха децата. А чувекъ го нема. Ако общината в града не вземе спешни мерки, ще се самуубеся ильи ще изчезна безкрайно."]

/Интервю с ромка излъчено по КАНАЛ 1 за акция на НСБОП по залавяне на опасни рецидивисти в Пазарджишко./

Enjoy!
Sanmayce 'Kaze', 2009 Mar 13.

```
D:\_KAZE_G.S._Corpus>yoshi
Yoshi(Filelist Creator), revision 06, written by Svalqyatchx,
in fact based on SWEEP.C from 'Open Watcom Project', thanks-thanks.

Note1: So far, it works for current directory only.
Note2: Default method is depth-first traversal;
       may use pipe 'Yoshi|sort' for breadth-first_like traversal results.
Note3: Make notice that '*.*'(extensionfull only) is not equal to '*'(all);
       one disadvantage is an inability to list only extensionless filenames.
Note4: Search is case-insensitive as-must.
Note5: This revision allows multiple '*', and meaning of masks is:
       '?' - any character AND NOT EMPTY(default, for OR EMPTY see option -e);
       '*' - any character(s) or empty.
Note6: What is a .LBL(LineByLine) file?
       it is a bunch of GRAMMATICAL lines not mere LF or CRLF lines;
       it contains not symbols under 32(except CR and LF) and above 127;
       it contains not space symbol sequences.
Usage:
      Yoshi [option(s)] [filename(s)]
      option(s):
         -v          i.e. verbose mode; output goes to console;
         -f          i.e. fullpath mode for output;
         -e          i.e. treat '?' as any character OR EMPTY;
         -t          i.e. touch all encountered files;
         -2          i.e. convert all encountered .TXT files to .LBL files;
         -o<filename> i.e. output goes to file(in append mode).
      filename(s):
         Wildcards '*' and wildcards '?' are allowed i.e. "str*.c??";
         default filename is '*'; DO NOT FORGET TO PUT
         filename(s) WITH WILDCARD(S) INTO QUOTE MARKS!
Examples:
      Yoshi -v -f -oCaterpillar_NON.lst "*.lbl" "*.txt" "*.htm" "*.html"
      Yoshi -f -oMyEbooks.txt "*wiley*essential*.pdf" "*russian*.*htm"

Yoshi: Total size of files: 00,027,750,342,332 bytes.
Yoshi: Total files: 000,000,001,088.
Yoshi: Total folders: 0,000,000,003.

D:\_KAZE_G.S._Corpus>"Leprechaun_r13++_32bits.exe"
Leprechaun(Fast Greedy Word-Ripper), revision 13++, written by Svalqyatchx.
Leprechaun: 'Oh, well, didn't you hear? Bigger is good, but jumbo is dear.'
Kaze: Let's see what a 4-way hash + 6,602,752 Binary-Search-Trees can give us,
      also the performance of a 4-way hash + 6,602,752 B-Trees of order 3.

'The Little Monster' short notes:
Note1: I wish to thank to R.N. Horspool, Ranjan Sinha, Dmitry Shkarin,
       Michael Abrash, J. Bentley, R. Sedgewick, Igor Pavlov, Lasse Reinhold
       for sharing their knowledge to public.
Note2: Run it without parameters to get usage and short notes.
Note3: This simple amateurish(more over I am not versed well neither in C nor
       in mathematics nor in english language, but I am persistent in INDEXING
       GBs of english TEXTS) tool is written in ANSI C(at least its source is
       compileable for CL(Windows) and GCC(Linux)), and its purpose is to
       create a WordList for a group of files(given via filelist).
       Its name comes(according to Heritage Dictionary) from 'low corpus' or
       'little body', in fact from amazing movie saga 'Leprechaun 1-2-3-4-5-6'
       starring by Warwick Davis.
Note4: Only words up to 31 chars are proceeded - the reason is 'DDT'(the
       longest word in Heritage Dictionary 3rd edition) or
       'dichlorodiphenyltrichloroethane'.
Note5: Cursor hiding in C - mission impossible for me.
Note6: By default(third parameter is 1023) allocated memory is 393MB.
       Due to 'malloc()' limitation under WINDOWS, maximum value of third
       parameter is 5174 which is 1988MB allocated block.
Note7: File Leprechaun.LOG is a log, where new statistics are appended.
Note8: Revision 12+ can handle files larger than 4GB.
Note9: Revision 12++ has a buffered 'fread()' - therefore I/O READ-BURST SPEED
       is the first(worst) bottleneck, as a result r.12++ is much-much faster;
       the second(worse) bottleneck: the linked lists - the b-trees
       might be the answer; the third(bad) bottleneck: the amateurish author.
NoteA: Revision 12+++ has an improved(2 bits were used doltishly) main hash
       function - therefore less collisions, for example:
       for file 'wikipedia-de-html.tar' 42,291,855,360 bytes with
       5,750,179,678 words of them 7,375,373 distinct attempts to Find/Put
       a WORD into a linked list are 6,117,675,470(r.12++) and 5,845,989,790
       (r.12+++); also two 'if' sections were moved because they were executed
       unnecessarily many times.
NoteB: Revision 13 uses BSTs instead of LLs, that is Linked-Lists were
       replaced by Binary-Search-Trees, as a result for 22,202,980 distinct
       words(out of 35,271,297) r.12+++ needs 225,548,268 total attempts to
       Find/Put WORDs into linked lists where r.13 needs 121,674,042 total
       attempts to Find/Put WORDs into Binary-Search-Trees. But this is a
       significant boost in performance only for wordlists of million words.
NoteC: Revision 13+ gives only more statistics. Future revisions could lessen
       number of attempts to Find/Put WORDs into Binary-Search-Trees
       furthermore by making them at some point Perfectly-Balanced. But
       for huge amount(multi-(m|b)illion) of distinct words the b-tree family
       must come in, until then this is the leprechaunish niche.
NoteD: Revision 13++ has a little fix(2 unnecessary ZEROings, when a new word
       is inserted, were deleted) and a fixed bug(13+ adds stupidly the
       highest BST to the wordlist). Also B-Tree of order 3 is added as a
       searching method. Main goal of B-Tree is to reduce number of
       comparisons but at nasty cost: a precious time wasted to construct it
       and twice more memory, i.e. one step forward two backward: this tree is
       more effective than BST in cases of 2++ billion/million
       different/distinct words.
       The improvement which comes from using B-Tree of order 3 is about 200%
       much more pleasing than I expected, for wikipedia-en-html.tar.wrd with
       12,561,874 distinct words Total Attempts to Find/Put WORDs into:
       Binary-Search-Trees was 61,895,043 while for
       B-trees order 3 was 19,295,791.
NoteE: For old r.12+ a USB connected HDD crippled test:
       for 'H:\>Leprechaun.exe static.wikipedia.org_downloads_2008-06_en.lst
       wikipedia-en-html.tar.wrd 5400'
       where 223,674,511,360 wikipedia-en-html.tar
       on laptop Toshiba Pentium T3400 2166 MHz with
       Motherboard Name:                            Toshiba Satellite L305
       CPU Type:         Mobile DualCore Intel Pentium, 2166 MHz (13 x 167)
       CPU Alias:                                                 Merom-1M
       L1 Code Cache:                                      32 KB per core
       L1 Data Cache:                                      32 KB per core
       L2 Cache:                        1 MB (On-Die, ECC, ASC, Full-Speed)
```

```
        Bus Type:                                                  Dual DDR2 SDRAM
        Bus Width:                                                         128-bit
        Real Clock:                                                  333 MHz (DDR)
        Effective Clock:                                                   666 MHz
        EVEREST v5.00.1650 Memory Copy:       3725MB/s with timings 5-5-5-13
        result is logged to 'Leprechaun.LOG':
   Bytes per second performance: 20,658,955B/s
   Words per second performance: 2,860,880W/s
   Input File with a list of TEXTual Files:
    static.wikipedia.org_downloads_2008-06_en.lst
   Size of all TEXTual Files: 223,674,511,360
   Word count: 30,974,750,142 of them 12,561,874 distinct
   Number Of Files: 1
   Number Of Lines: 2088618575
   Allocated memory in MB: 1920
   Words with length 01 occupy 0,033KB of 0,349KB given i.e. 09% utilization
   Words with length 02 occupy 0,033KB of 0,349KB given i.e. 09% utilization
   Words with length 03 occupy 0,037KB of 0,697KB given i.e. 05% utilization
   Words with length 04 occupy 0,151KB of 0,871KB given i.e. 17% utilization
   Words with length 05 occupy 0,744KB of 1,568KB given i.e. 47% utilization
   Words with length 06 occupy 1,470KB of 3,136KB given i.e. 46% utilization
   Words with length 07 occupy 2,605KB of 5,923KB given i.e. 43% utilization
   Words with length 08 occupy 3,296KB of 6,968KB given i.e. 47% utilization
   Words with length 09 occupy 3,714KB of 6,968KB given i.e. 53% utilization
   Words with length 10 occupy 3,483KB of 6,968KB given i.e. 49% utilization
   Words with length 11 occupy 3,235KB of 5,923KB given i.e. 54% utilization
   Words with length 12 occupy 2,691KB of 4,181KB given i.e. 64% utilization
   Words with length 13 occupy 2,230KB of 3,484KB given i.e. 64% utilization
   Words with length 14 occupy 1,718KB of 3,484KB given i.e. 49% utilization
   Words with length 15 occupy 1,357KB of 2,613KB given i.e. 51% utilization
   Words with length 16 occupy 1,063KB of 2,613KB given i.e. 40% utilization
   Words with length 17 occupy 0,814KB of 1,742KB given i.e. 46% utilization
   Words with length 18 occupy 0,617KB of 1,742KB given i.e. 35% utilization
   Words with length 19 occupy 0,485KB of 1,742KB given i.e. 27% utilization
   Words with length 20 occupy 0,402KB of 1,742KB given i.e. 23% utilization
   Words with length 21 occupy 0,327KB of 1,742KB given i.e. 18% utilization
   Words with length 22 occupy 0,274KB of 1,742KB given i.e. 15% utilization
   Words with length 23 occupy 0,224KB of 1,394KB given i.e. 16% utilization
   Words with length 24 occupy 0,190KB of 1,394KB given i.e. 13% utilization
   Words with length 25 occupy 0,162KB of 1,394KB given i.e. 11% utilization
   Words with length 26 occupy 0,136KB of 1,220KB given i.e. 11% utilization
   Words with length 27 occupy 0,119KB of 1,046KB given i.e. 11% utilization
   Words with length 28 occupy 0,107KB of 0,871KB given i.e. 12% utilization
   Words with length 29 occupy 0,091KB of 0,697KB given i.e. 13% utilization
   Words with length 30 occupy 0,080KB of 0,523KB given i.e. 15% utilization
   Words with length 31 occupy 0,076KB of 0,523KB given i.e. 14% utilization
   Total pseudo(including hash table) memory utilization: 42%
   Total real(wordlist's words VS allocated block) memory utilization: 60/1000
   Used value for third parameter in KB: 5400
   Use next time as third parameter: 3475-
   Time for making unsorted wordlist: 10827 second(s)
   Time for sorting unsorted wordlist: 10 second(s)

Usage: Leprechaun InFile OutFile [BufferSize] [SortMethod] [TreeMethod]
      <InFile>: Input file with files for Leprechauning, in WINDOWS console
                you can create it by 'E:\KAZEHOME>dir *.txt/s/b>Leprechaun.lst'
      <OutFile>: Output WORDLIST(sorted since r.9, CRLF) file
      <BufferSize>: Optional Dynamic RAM buffer in KB, default(and minimum
                    in the same time) is 1023, i.e. omit or specify greater one
      <SortMethod>: Optional Sort Method, default is 'D',
                    A - InsertionSort
                    B - InsertionX26Sort
                    C - MultiKeyQuickSortSort by J. Bentley, R. Sedgewick
                    D - MultiKeyQuickSortX26Sort' by J. Bentley, R. Sedgewick
      <TreeMethod>: Optional Tree Method, default is 'X',
                    X - Binary-Search-Trees
                    Y - B-Trees of order 3

Have a nice Leprechauning.
For contacts: sanmayce@hotmail.com
Sanmayce Svalqyatchx 'Kaze', 2005 Feb 07(rev.13++: 2010 Apr 12).
```

D:_KAZE_G.S._Corpus>Caterpillar

```
Caterpillar(Sentence_Dumper), revision 14+, written by Svalqyatchx,
in fact adapted from Haruhiko Okumura's excellent LZSS.C program.

       How near are these words_forms to me: Masakari, Massacre, Steel-Coloss,
Monster-Truck, Dump-Mining-Truck, Caterpillar 797, Liebherr, Komatsu. They
resemble one thing: strong-devoid-of-ambition-power(i.e. a pure work/time).

       'Caterpillar' is a simple pattern searcher(from 'Masakari' family tools)
into archived english-text files, designed to achieve up to 90% higher read
speed than the HDD READ BURST i.e. 'copy hugefile nul' gaining at same time
50% compression of searched data.

       Its main feature is somewhat hidden nowadays, because of
pseudo-transparent decompression used, which leads to doubling(unreachable in
fact) uploaded data for search function(written by N. Horspool, thanks a lot)
due to LZSS algorithm implemented by H. Okumura(greetings to him). Okumura's
variant(HDD2RAM) which is much faster(!!!) and needs less memory than tuned
memory-to-memory decompression(RAM2RAM) variant. I am still stunned.

       In few words: feeding search function is 100-% faster with very fast
CPU-Physical_RAM subsystems, in this way reducing the ugly penalty which comes
from reading a HDD. In numbers: me IDE HITACHI 7200rpm 2MB gives up to 60MB/s
READ BURST, 'Caterpillar' almost doubles(i.e. 120-MB/s) it in case of 3+++GHz
CPU and 533+++MHz RAM.

       For Windows 2003, VIA KT600, AMD XP 2500+(1836.12MHz=11x166.92MHz),
FSB 333.84MHz(2x166.92MHz), 512KB L2 cache, 1 DIMM DDR 512MB 333MHz(2x166MHz),
Caterpillar(in fact LZSS) decompresses 58,000KB per second i.e.
boost is negative: 60MB/s=61,440KB/s(READ BURST) is greater than 58,000KB/s.
But for two times faster CPU-RAM sub-system(SERVER) than described above OR
for two times slower HDD sub-system(LAPTOP) boost will be positive:
(1 - READ BURST SPEED / DECOMPRESSION SPEED) * READ BURST SPEED or
(1 - (61,440KB/s) / (2 * 58,000KB/s)) * 61,440KB/s = (0.471) * 61,440KB/s.

       Since revision 5 'fread()' was changed with 'read()', for speed.

'The Monster-Dump-Truck' short notes:
Note1: Thanks a lot to N. Horspool, Dmitry Shkarin, H. Okumura, Igor Pavlov.
Note2: Run it without parameters to get usage and short notes.
```

```
Note3: Current revision searches only for case-sensitive and unexact matches.
Note4: This simple amateurish(more over I am not versed well neither in C nor
       in mathematics nor in english language, but I am persistent in INDEXING
       GBs of english TEXTS) tool is written in ANSI C(at least its source is
       compileable for CL(Windows) and not yet for GCC(Linux) because of
       'O_BINARY in open(), gets(), getch(), kbhit()', and its purpose is to
       create a SentenceList for a group of compressed(with it) text files(LF
       and CRLF) given via filelist.
       Its name comes from a heavy-nopride-dumper-truck 'Caterpillar'.
Note5: By default allocated memory is 95MB i.e. decoding is HDD2RAM.
Note6: Disastrous performance in case 95MB|147MB not fully physical!
Note7: For me digital library:
       where files are 54, ENcoded 6,917,425,566, DEcoded 14,419,485,826
       with Windows XP, VIA KT600, AMD XP 2500+(1836.12MHz=11x166.92MHz),
       FSB 333.84MHz(2x166.92MHz), 512KB L2 cache, DDR 512MB 333MHz(2x166MHz),
       IDE HDD Maxtor 80GB 7200 8MB and
       'D:\temp>dir E:\KAZEHOME\KAZUYA.O??/b>Caterpillar.lst'
       'D:\temp>Caterpillar Caterpillar.lst CaterpillarRAM2RAM.ini'
       result is: 282 seconds or 41000KB/s upload, 11000KB/s boost,
       52000KB/s boosted upload, 56000KB/s decode.
       'D:\temp>Caterpillar Caterpillar.lst CaterpillarHDD2RAM.ini'
       result is: 142 seconds or 99000KB/s boosted upload!!!
Note8: Matches(hits) containing neither '<' nor '>' are written
       to 'Caterpillar.hits.pattern?.html' file.
Note9: Works both on UNIX(LF) and Windows(CRLF) text files.
NoteA: Never forget the importance of defragmented_AND_grouped files located at
       fastest area of disk - first partition is faster than second one, etc.
NoteB: In ANSI, clock is defined as '#define CLOCKS_PER_SEC 1000'.
NoteC: Since Caterpillar 13++:
       - limits(just skip longer ones) lines to 960 chars; OTHERWISE: HUGE TIME
         DELAYS due to recursive function;
       - shows hits to console too; MORE VIVID;
NoteD: During execution hitting a 'Esc' causes termination(i.e. skipping rest).
NoteE: At last NON-ENCODED regime has two modes: in addition to LINE(i.e.
       hits are lines) there is a FILE(i.e. hits are filenames) mode.
NoteF: For all regimes files Caterpillar.HIT?.lst are created for each
       pattern(1,2,3 and 4) - containing hits filelist i.e. filenames
       containing HITS(either LINEs or FILENAMEs).

Below is LINE(default for DECODING ???2RAM regimes) mode pattern description:
Pattern(s) note: You may specify(four times) a main-pattern(case insensitive
                 with wildcards '*' i.e. any character(s) or empty and '?'
                 i.e. any character or empty) with three nested-patterns(case
                 sensitive and unexact), all four connected with AND.
                 Due to different line endings(CRLF in Windows; LF in UNIX)
                 you must add a '?' wildcard in place of CR: for example in
                 case of searching for '*.pdf' write '*.pdf?'.
Pattern(s) example: Pattern1: *take? *it*
                    Pattern1_NestedPattern1: you
                    Possible hit: ... your reason is so taken by It.

Usage: 'Caterpillar e file1 file2' encodes file1 into file2
       'Caterpillar d file2 file1' decodes file2 into file1
       'Caterpillar m ListOfFilesFile SolidSize'
         <ListOfFilesFile>: Files to be merged into Caterpillar.??? files
         <SolidSize>: Caterpillar.??? files size limit in MB.
       'Caterpillar ListOfFilesFile [OptionsFile]'
         <ListOfFilesFile>: Input file with files for Caterpillaring
         <OptionsFile>: Optional input file with options with following format:
         Optional line #1 contains method of decoding:
           'DECODING HDD2RAM' | 'DECODING RAM2RAM' | 'NON-ENCODED'
           'NON-ENCODED' allocates 95MB, size of biggest file must be lower;
           'DECODING HDD2RAM' needs less physical memory(95MB) but is faster!
           'DECODING RAM2RAM' needs more physical memory(147MB) but is slower!
         Optional line #2 contains terminal hits:
           '0' | 'long integer'
           '0' means all hits are needed
           'long integer' means reaching this value termination follows
         Optional line #3 contains Pattern1: 'string'
           if 'string' is specified then input from keyboard arise not
           if 'string' is not specified then input from keyboard arise
         Optional line #4 contains Pattern1_NestedPattern1: 'string'
         Optional line #5 contains Pattern1_NestedPattern2: 'string'
         Optional line #5 contains Pattern1_NestedPattern3: 'string'
Note1: One useful way to make 'ListOfFilesFile=Caterpillar_NON.lst' is next:
D:\Caterpillar>copy con MAKElst.bat
@echo off
dir Caterpillar_tree\*.lbl /s/b>Caterpillar_NON.lst
dir Caterpillar_tree\*.txt /s/b>>Caterpillar_NON.lst
echo.
F6

Have a nice Caterpillaring.
For contacts: sanmayce@hotmail.com
Sanmayce Svalqyatchx 'Kaze', 2009 Jan 29.
```

D:_KAZE_G.S._Corpus>Raccoondog.exe

```
LZMA Utility 4.65 : Igor Pavlov : Public domain : 2009-02-03

Usage:  lzma <e|d> inputFile outputFile
  e: encode file
  d: decode file
```

D:_KAZE_G.S._Corpus>Raccoondog -SA4 Raccoondog.lst

```
Raccoondog(LZMA Sentence_Dumper), revision 17++, written by Svalqyatchx,
in fact adapted from Igor Pavlov's excellent LZMA 4.56 SDK.

Usage1: Raccoondog [-SA1|-SA2|-SA3|-SA4] filename
        Decodes all files from a list(filename)
        -SA1 : Brute_Force Search Algorithm
        -SA2 : Quick_Boyer_Moore Search Algorithm
        -SA3 : SMITH_Boyer_Moore Search Algorithm
        -SA4 : Karp_Rabin_Kaze Search Algorithm
        Default is HORSPOOL_Boyer_Moore Search Algorithm
Usage2: Raccoondog <e|d> inputFile outputFile
        e: encode file
        d: decode file
Example1: Raccoondog Raccoondog.lst
Example2: Raccoondog -SA2 Raccoondog.lst
Example3: Raccoondog e Caterpillar.001.txt Caterpillar.001.txt.lzma
Note1: Benchmark:
```

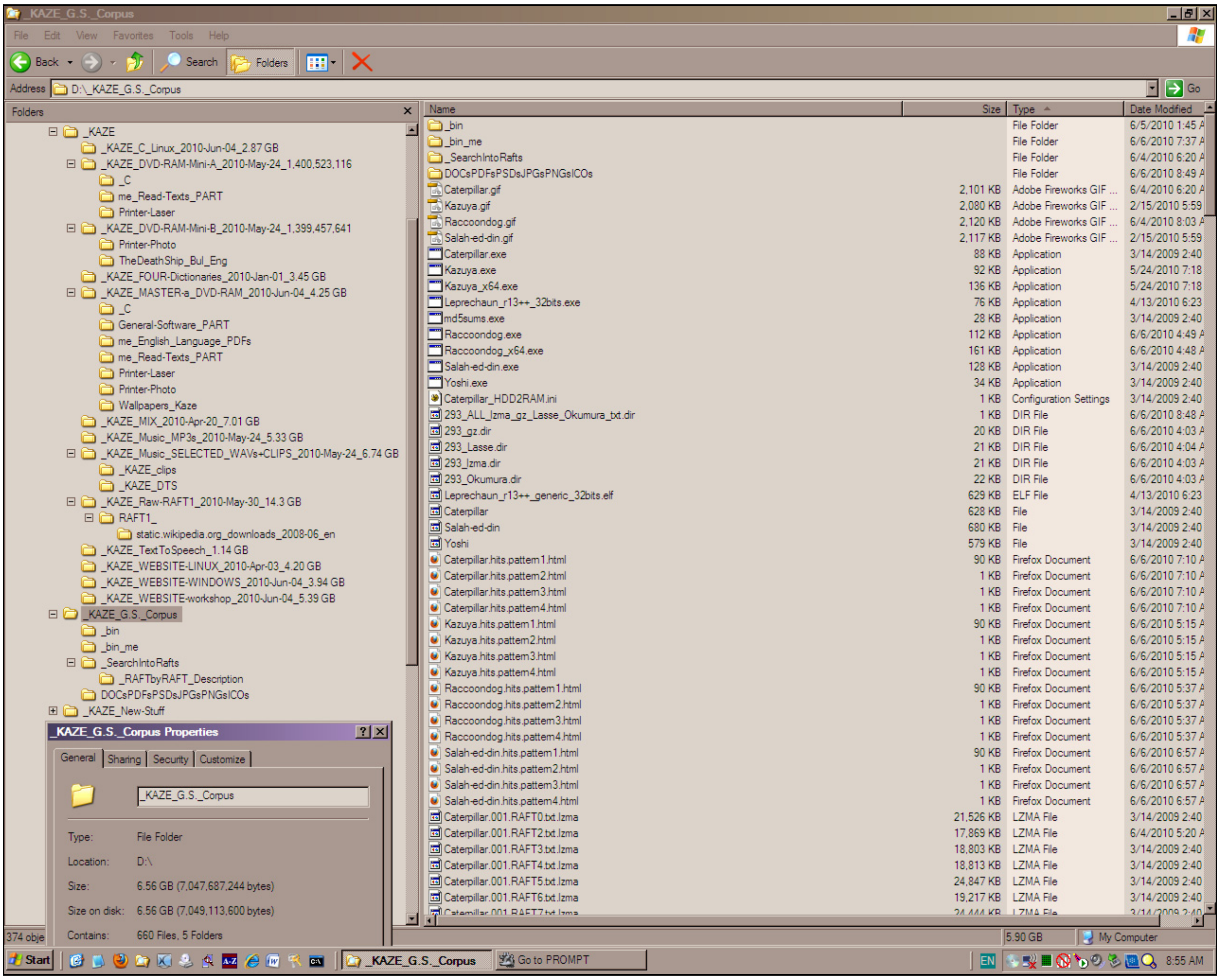
```
       Raccoondog(EN:8KB/clock, DE:39KB/clock) for 24.9GB(5.53GB LZMA) texts.
       Me machine is:
       Motherboard Name:                             Toshiba Satellite L305
       CPU Type:          Mobile DualCore Intel Pentium, 2166 MHz (13 x 167)
       CPU Alias:                                                  Merom-1M
       L1 Code Cache:                                         32 KB per core
       L1 Data Cache:                                         32 KB per core
       L2 Cache:                        1 MB (On-Die, ECC, ASC, Full-Speed)
       Bus Type:                                            Dual DDR2 SDRAM
       Bus Width:                                                   128-bit
       Real Clock:                                            333 MHz (DDR)
       Effective Clock:                                             666 MHz
Note2: Disastrous performance in case 128MB not fully physical!
Note3: Matches(hits) are overwritten to Raccoondog.hits.Pattern?.html files.
Note4: Works both on UNIX(LF) and Windows(CRLF) text files.
Note5: Never forget the importance of defragmented_AND_grouped files located at
       fastest area of disk - first partition is faster than second one, etc.
Note6: In ANSI, clock is defined as '#define CLOCKS_PER_SEC 1000'.
Note7: Since Raccoondog 13++:
       - limits(just skip longer ones) lines to 960 chars; OTHERWISE: HUGE TIME
         DELAYS due to recursive function;
       - shows hits to console too; MORE VIVID;
Note8: Since Raccoondog 14:
       - No deletion of input file after compressing/decompressing;
Note9: During execution hitting a 'Esc' causes termination(i.e. skipping rest).
NoteA: The two examples below show the need of one additional wildcard in
       order to match CR for Windows texts; end of line is LF(as in UNIX):
       Pattern(s) example: Pattern1: ########%
                           Pattern1_NestedPattern1:
                           Possible hit: NEW YORK
       Pattern(s) example: Pattern1: $$$$$$$$@
                           Pattern1_NestedPattern1:
                           Possible hit: Printing

Pattern(s) note: You may specify(four times) a main-pattern(case insensitive
                 with wildcards '*' i.e. any character(s) or empty, also '@'
                 i.e. any character or empty, also '#' i.e. any character
                 and not empty, also '$' i.e. any ALPHA character
                 and not empty, also '%' i.e. any NON-ALPHA character
                 and not empty) with three nested-patterns(case
                 sensitive and unexact), all four connected with AND.
                 Due to different line endings(CRLF in Windows; LF in UNIX)
                 you must add a '@' wildcard in place of CR: for example in
                 case of searching for '*.pdf' write '*.pdf@'.
Pattern(s) example: Pattern1: *%take@%$$@
                    Pattern1_NestedPattern1:
                    Possible hit: ... is taken by
                    Possible hit: ... would take it
                    Note: First % is to avoid e.g. 'mis' prefix
                          Second % is to avoid e.g. 'ing' suffix
Master-pattern note: It is case insensitive with wildcards '*','@','#','$','%'
                     allowed. The purpose of this pattern is to
                     decide whether a search for next patterns will be
                     executed, it is applied on all lines i.e. the whole file.
                     There must be at least one hit in order to execute search
                     for next patterns.

Have a nice Raccoondoging.
For contacts: sanmayce@hotmail.com
Sanmayce Svalqyatchx 'Kaze', 2010 Jun 06.

Allocated memory for DEcoded file in MB: 256
Size of input file with files for Raccoondoging: 9669

Input Master-pattern(hit only 'Enter' to skip):
Input Pattern1(hit only 'Enter' to skip): *not anymore*
- Input Pattern1_NestedPattern1(hit only 'Enter' to skip):
Input Pattern2(hit only 'Enter' to skip):
Processing .\Caterpillar.001.RAFT2.txt.lzma ...
Doing DECODE from HDD to RAM ...
Overall decode performance so far: 000,007KB/clock(EN) or 000,031KB/clock(DE)
Doing SEARCH for Pattern1 at once and flushing hit-sentences ...
000,000,001 It used to be just "the living room," but not anymore.
000,000,002 But not anymore.
Found 2 case-insensitive and unexact matches(hits), so far.
'Esc' was pressed, so skip the rest files and quit!

Total Rough Upload and Decode time: 2,297 clocks
Total Rough Search time: 1,719 clocks
Total time: 4 seconds
Total Lines encountered: 1,150,388
Total Search(non-mask) function invocations: 0
Total Search(MASK i.e. wildcard) function invocations: 1,149,075
Total MASK i.e. wildcard invocations: 1,149,075
Total MASK i.e. wildcard hits: 2
Total MASK i.e. wildcard time: 1,434 clocks
Total MASK i.e. wildcard performance: 46KB/clock
Total BoyerMooreHorspool invocations: 0
Total BoyerMooreHorspool(whole chunks, not lines) hits: 0
Total BoyerMooreHorspool(whole chunks, not lines) time: 0 clocks
Total KarpRabinKaze invocations: 0
Total KarpRabinKaze(whole chunks, not lines) hits: 0
Total KarpRabinKaze(whole chunks, not lines) time: 0 clocks
Raccoondog: Done successfully.

D:\_KAZE_G.S._Corpus>Salah-ed-din -SA4 Salah-ed-din.lst
Salah-ed-din(Sentence_Dumper), revision 14++, written by Svalqyatchx,
in fact adapted from Mark Adler's and Jean-loup Gailly's ZLIB package.

Usage1: Salah-ed-din [-SA1|-SA2|-SA3|-SA4] filename
        Decodes all files from a list(filename)
        -SA1 : Brute_Force Search Algorithm
        -SA2 : Quick_Boyer_Moore Search Algorithm
        -SA3 : SMITH_Boyer_Moore Search Algorithm
        -SA4 : Karp_Rabin_Kaze Search Algorithm
        Default is HORSPOOL_Boyer_Moore Search Algorithm
Usage2: Salah-ed-din [-d] [-f] [-h] [-r] [-1 to -9] [files...]
        -d : decompress
        -f : compress with Z_FILTERED
        -h : compress with Z_HUFFMAN_ONLY
        -r : compress with Z_RLE
        -1 to -9 : compression level
```

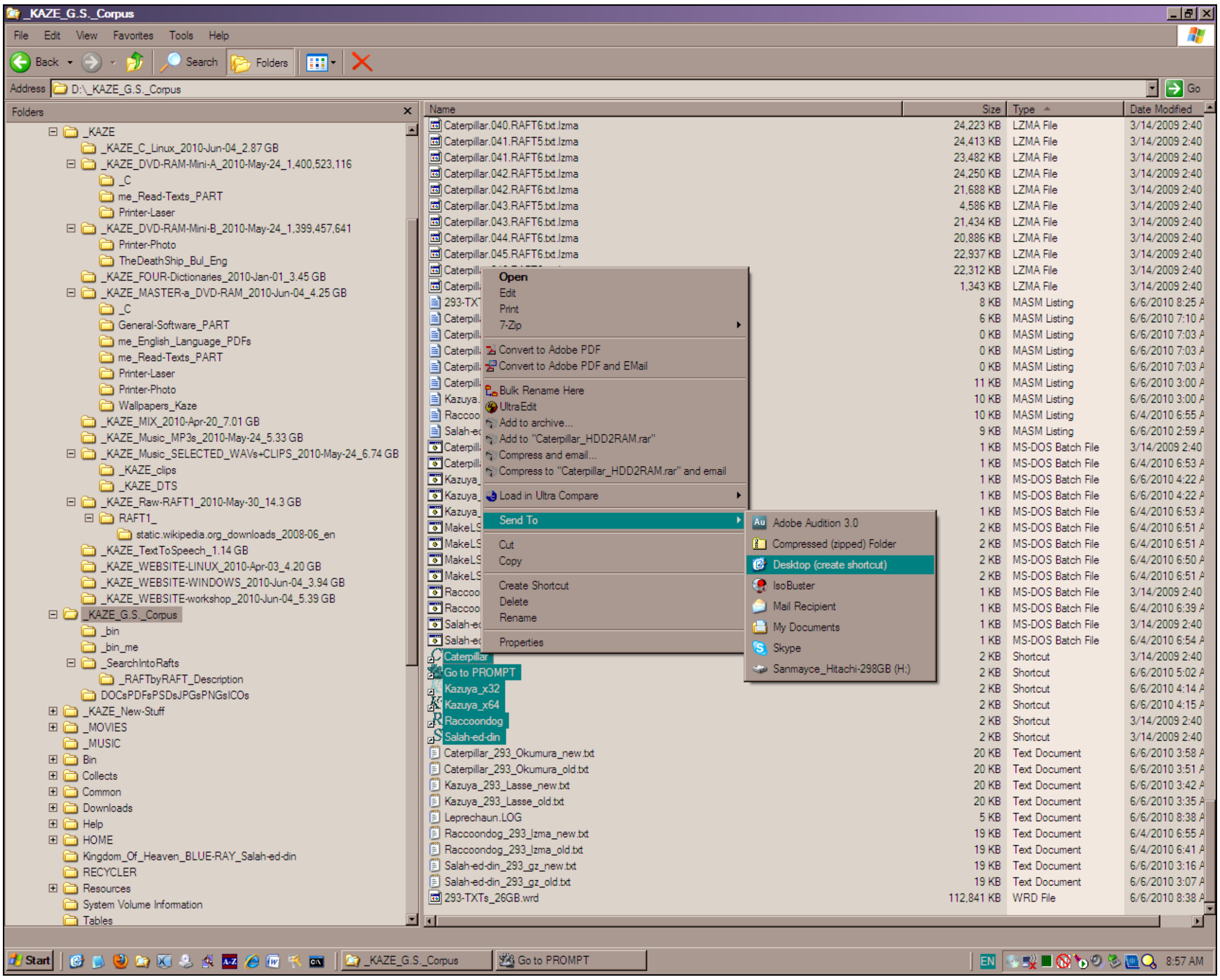
```
Example1: Salah-ed-din Salah-ed-din.lst
Example2: Salah-ed-din -SA2 Salah-ed-din.lst
Example3: Salah-ed-din -f -6 Caterpillar.001.txt
Example4: Salah-ed-din -d Caterpillar.001.txt.gz
Note1: Benchmark:
       Raccoondog(EN:39KB/clock, DE:117KB/clock) for 24.9GB(8.42GB GZ) texts.
       Me machine is:
       Motherboard Name:                               Toshiba Satellite L305
       CPU Type:         Mobile DualCore Intel Pentium, 2166 MHz (13 x 167)
       CPU Alias:                                                    Merom-1M
       L1 Code Cache:                                        32 KB per core
       L1 Data Cache:                                        32 KB per core
       L2 Cache:                        1 MB (On-Die, ECC, ASC, Full-Speed)
       Bus Type:                                            Dual DDR2 SDRAM
       Bus Width:                                                    128-bit
       Real Clock:                                             333 MHz (DDR)
       Effective Clock:                                              666 MHz
Note2: Disastrous performance in case 128MB not fully physical!
Note3: Matches(hits) are overwritten to Salah-ed-din.hits.Pattern?.html files.
Note4: Works both on UNIX(LF) and Windows(CRLF) text files.
Note5: Never forget the importance of defragmented_AND_grouped files located at
       fastest area of disk - first partition is faster than second one, etc.
Note6: In ANSI, clock is defined as '#define CLOCKS_PER_SEC 1000'.
Note7: Since Salah-ed-din 13++:
       - limits(just skip longer ones) lines to 960 chars; OTHERWISE: HUGE TIME
         DELAYS due to recursive function;
       - shows hits to console too; MORE VIVID;
Note8: Since Salah-ed-din 14:
       - No deletion of input file after compressing/decompressing;
Note9: During execution hitting a 'Esc' causes termination(i.e. skipping rest).

Pattern(s) note: You may specify(four times) a main-pattern(case insensitive
                 with wildcards '*' i.e. any character(s) or empty and '?'
                 i.e. any character or empty) with three nested-patterns(case
                 sensitive and unexact), all four connected with AND.
                 Due to different line endings(CRLF in Windows; LF in UNIX)
                 you must add a '?' wildcard in place of CR: for example in
                 case of searching for '*.pdf' write '*.pdf?'.
Pattern(s) example: Pattern1: *take? *it*
                    Pattern1_NestedPattern1: you
                    Possible hit: ... your reason is so taken by It.

Have a nice Salah-ed-dining.
For contacts: sanmayce@hotmail.com
Sanmayce Svalqyatchx 'Kaze', 2009 Feb 22.

Allocated memory for DEcoded file in MB: 96
Size of input file with files for Salah-ed-dining: 8680
Pattern(s) note: You may specify(four times) a main-pattern(case insensitive
                 with wildcards '*' i.e. any character(s) or empty and '?'
                 i.e. any character or empty) with three nested-patterns(case
                 sensitive and unexact), all four connected with AND.
                 Due to different line endings(CRLF in Windows; LF in UNIX)
                 you must add a '?' wildcard in place of CR: for example in
                 case of searching for '*.pdf' write '*.pdf?'.
Pattern(s) example: Pattern1: *take? *it*
                    Pattern1_NestedPattern1: you
                    Possible hit: ... your reason is so taken by It.

Input Pattern1(hit only 'Enter' to skip): *not anymore*
- Input Pattern1_NestedPattern1(hit only 'Enter' to skip):
Input Pattern2(hit only 'Enter' to skip):
Processing .\Caterpillar.001.RAFT3.txt.gz ...
Doing DECODE from HDD to RAM ...

Salah-ed-din decoded buffer size: 99,614,459
Overall decode performance so far: 000,033KB/clock(EN) or 000,102KB/clock(DE)
Doing SEARCH for Pattern1 at once and flushing hit-sentences ...
000,000,001 M: Not anymore.
000,000,002 M: Not anymore.
000,000,003 "Not anymore," Lidia replied.
000,000,004 Not anymore.
000,000,005 Not anymore.
000,000,006 "Not anymore," Lidia replied.
000,000,007 Not anymore.
000,000,008 Not anymore.
000,000,009 "Not anymore," Lidia replied.
000,000,010 Not anymore.
000,000,011 Not anymore.
Found 11 case-insensitive and unexact matches(hits), so far.
'Esc' was pressed, so skip the rest files and quit!

Total Rough Upload and Decode time: 953 clocks
Total Rough Search time: 1,907 clocks
Total time: 3 seconds
Total Lines encountered: 1,835,098
Total Search(non-mask) function invocations: 0
Total Search(MASK i.e. wildcard) function invocations: 1,834,650
Total Boyer-Moore-Horspool(whole chunks, not lines) hits: 0
Total Boyer-Moore-Horspool(whole chunks, not lines) time: 0 clocks
Total Karp_Rabin_Kaze(whole chunks, not lines) hits: 0
Total Karp_Rabin_Kaze(whole chunks, not lines) time: 0 clocks
Salah-ed-din: Done successfully.
```

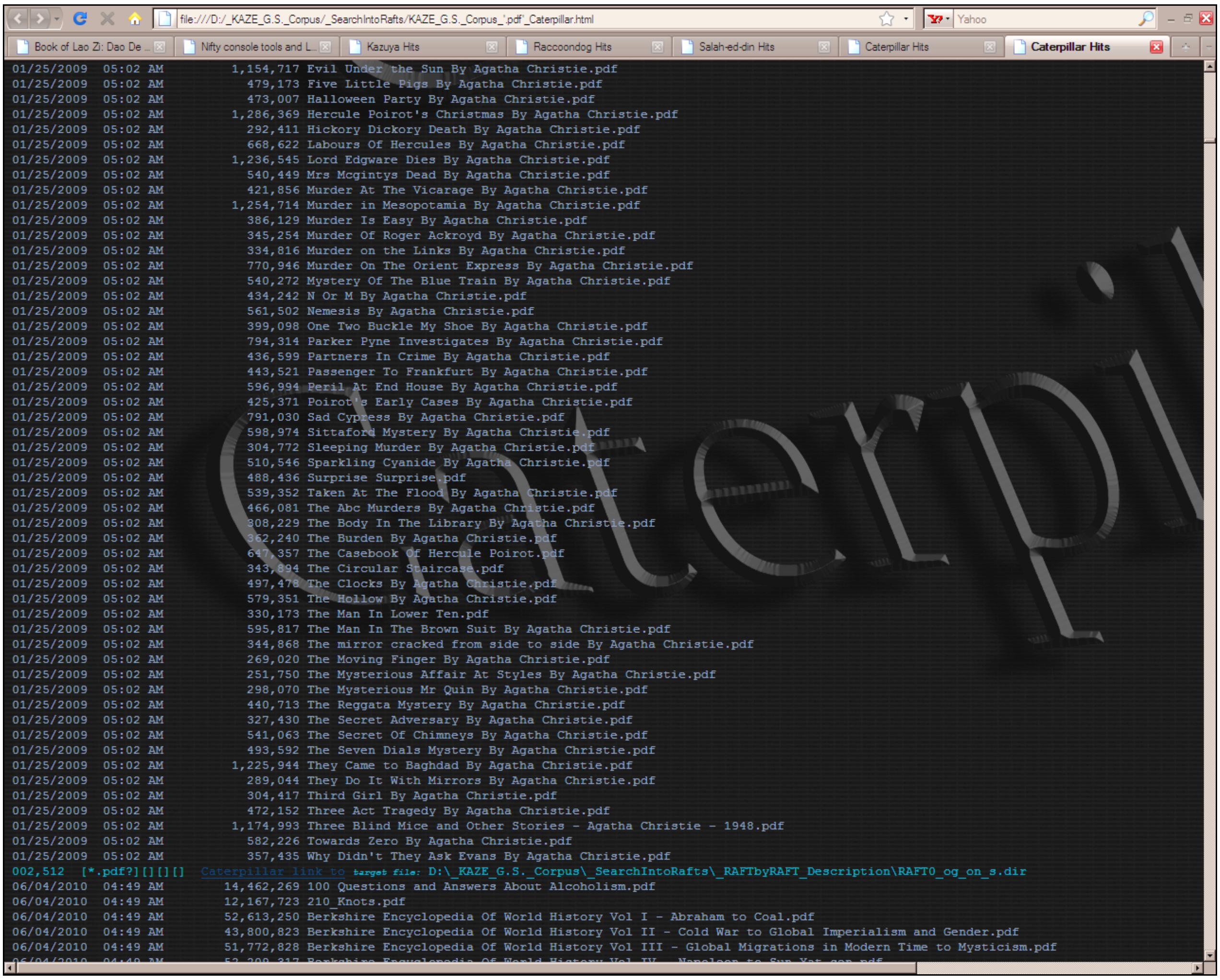
D:_KAZE_G.S._Corpus>Kazuya.exe/?

```
Kazuya(LZ Sentence_Dumper), revision 17++, written by Svalqyatchx,
in fact adapted from Lasse Reinhold's excellent QuickLZ 1.4.0 library,
in fact adapted from Ariya Hidayat's sub-excellent FastLZ 0.1.0 library,
in fact adapted from Markus F.X.J. Oberhumer's sub-excellent LZO 2.03 library,
in fact adapted from Haruhiko Okumura's sub-excellent LZSS 4/6/1989 library.

Usage1: Kazuya [-sa1|-sa2|-sa3|-sa4|-SA1|-SA2|-SA3|-SA4
               |-sA1|-sA2|-sA3|-sA4|-Sa1|-Sa2|-Sa3|-Sa4|-krknd] filename
        Decodes all files from a list(filename)
        -sa1 : QuickLZ Decode + Brute_Force Search Algorithm
        -sa2 : QuickLZ Decode + Quick_Boyer_Moore Search Algorithm
        -sa3 : QuickLZ Decode + SMITH_Boyer_Moore Search Algorithm
        -sa4 : QuickLZ Decode + Karp_Rabin_Kaze Search Algorithm
        -SA1 : LZO Decode + Brute_Force Search Algorithm
        -SA2 : LZO Decode + Quick_Boyer_Moore Search Algorithm
        -SA3 : LZO Decode + SMITH_Boyer_Moore Search Algorithm
        -SA4 : LZO Decode + Karp_Rabin_Kaze Search Algorithm
```

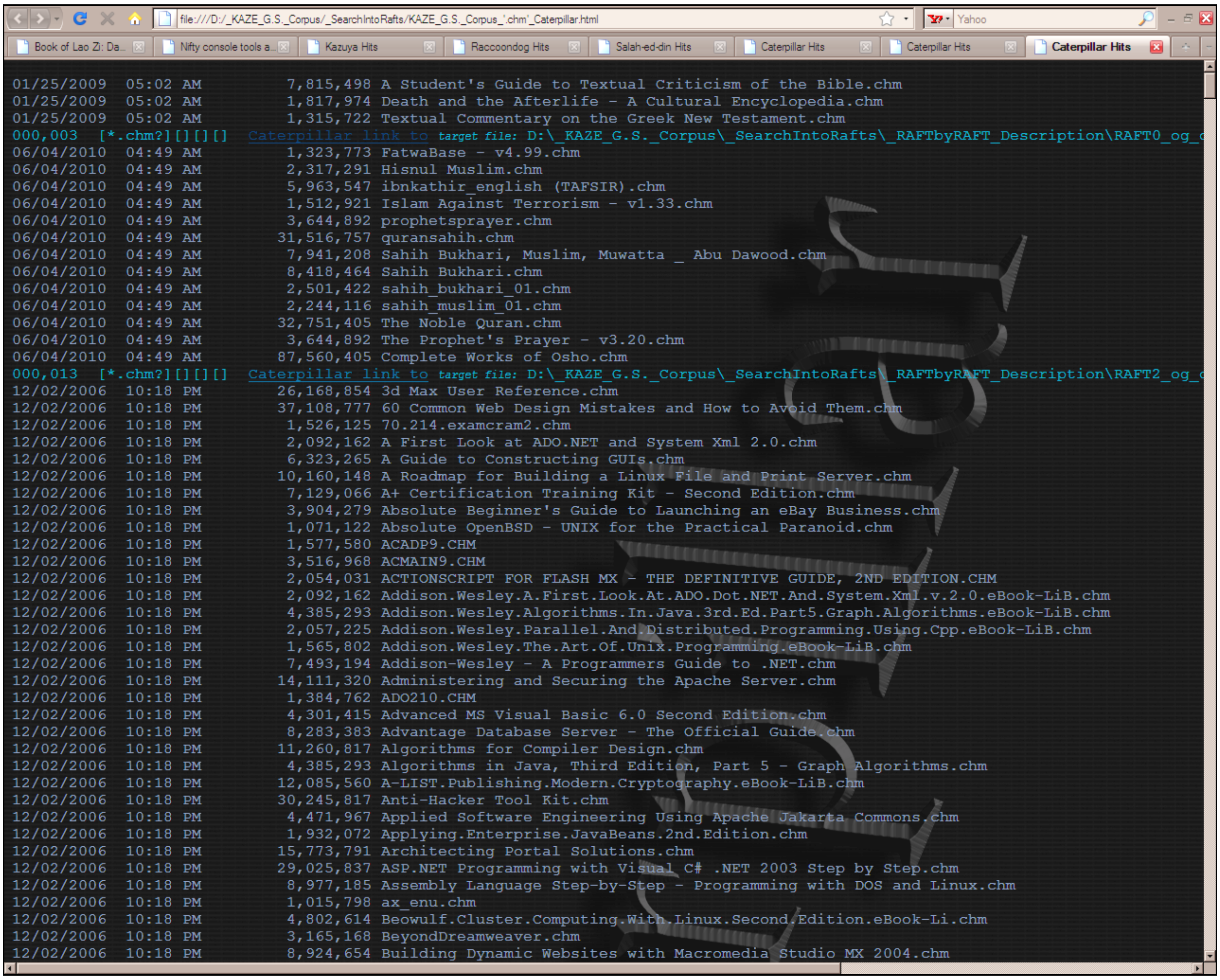
```
        -sA1 : FastLZ Decode + Brute_Force Search Algorithm
        -sA2 : FastLZ Decode + Quick_Boyer_Moore Search Algorithm
        -sA3 : FastLZ Decode + SMITH_Boyer_Moore Search Algorithm
        -sA4 : FastLZ Decode + Karp_Rabin_Kaze Search Algorithm
        -Sa1 : OkumuraLZ Decode + Brute_Force Search Algorithm
        -Sa2 : OkumuraLZ Decode + Quick_Boyer_Moore Search Algorithm
        -Sa3 : OkumuraLZ Decode + SMITH_Boyer_Moore Search Algorithm
        -Sa4 : OkumuraLZ Decode + Karp_Rabin_Kaze Search Algorithm
        -krknd : Karp_Rabin_Kaze Search Algorithm, but with no additional
                 chunk-searches overhead and without decompression of
                 incoming files i.e. pure text is uploaded
        Default is QuickLZ Decode + HORSPOOL_Boyer_Moore Search Algorithm
Usage2: Kazuya <e|d|E|D|A|R|a|r> inputFile outputFile
        e: encode QuickLZ file
        d: decode QuickLZ file
        E: encode LZO file
        D: decode LZO file
        A: encode(archive) FastLZ file
        R: decode(restore) FastLZ file
        a: encode(archive) OkumuraLZ file
        r: decode(restore) OkumuraLZ file
Example1: Kazuya Kazuya.lst
Example2: Kazuya -SA2 Kazuya.lst
Example3: Kazuya e Caterpillar.001.txt Caterpillar.001.txt.Lasse
Note1: Benchmark(HDD read speed is the nasty bottleneck):
       Kazuya(EN:37KB/clock, DE:88KB/clock) for 24.9GB(10.6GB Lasse) texts.
       Me machine is:
       Motherboard Name:                             Toshiba Satellite L305
       CPU Type:         Mobile DualCore Intel Pentium, 2166 MHz (13 x 167)
       CPU Alias:                                                  Merom-1M
       L1 Code Cache:                                      32 KB per core
       L1 Data Cache:                                      32 KB per core
       L2 Cache:                     1 MB (On-Die, ECC, ASC, Full-Speed)
       Bus Type:                                         Dual DDR2 SDRAM
       Bus Width:                                                 128-bit
       Real Clock:                                          333 MHz (DDR)
       Effective Clock:                                           666 MHz
Note2: Disastrous performance in case 256MB not fully physical!
Note3: Matches(hits) are overwritten to Kazuya.hits.Pattern?.html files.
Note4: Works both on UNIX(LF) and Windows(CRLF) text files.
Note5: Never forget the importance of defragmented_AND_grouped files located at
       fastest area of disk - first partition is faster than second one, etc.
Note6: In ANSI, clock is defined as '#define CLOCKS_PER_SEC 1000'.
Note7: Since Kazuya 13++:
       - limits(just skip longer ones) lines to 960 chars; OTHERWISE: HUGE TIME
         DELAYS due to recursive function;
       - shows hits to console too; MORE VIVID;
Note8: Since Kazuya 14:
       - No deletion of input file after compressing/decompressing;
Note9: During execution hitting a 'Esc' causes termination(i.e. skipping rest).
NoteA: This revision works with up to 127MB incoming(non-compressed) files,
       because it allocates 256MB of which one half is for incoming
       oher for outcoming file, i.e. decompression/compression is RAM to RAM.
NoteB: Charge(delivery) performance combines upload and decode performance.
NoteC: The two examples below show the need of one additional wildcard in
       order to match CR for Windows texts; end of line is LF(as in UNIX):
       Pattern(s) example: Pattern1: ########%
                           Pattern1_NestedPattern1:
                           Possible hit: NEW YORK
       Pattern(s) example: Pattern1: $$$$$$$$@
                           Pattern1_NestedPattern1:
                           Possible hit: Printing

Pattern(s) note: You may specify(four times) a main-pattern(case insensitive
                 with wildcards '*' i.e. any character(s) or empty, also '@'
                 i.e. any character or empty, also '#' i.e. any character
                 and not empty, also '$' i.e. any ALPHA character
                 and not empty, also '%' i.e. any NON-ALPHA character
                 and not empty) with three nested-patterns(case
                 sensitive and unexact), all four connected with AND.
                 Due to different line endings(CRLF in Windows; LF in UNIX)
                 you must add a '@' wildcard in place of CR: for example in
                 case of searching for '*.pdf' write '*.pdf@'.
Pattern(s) example: Pattern1: *%take@%$$@
                    Pattern1_NestedPattern1:
                    Possible hit: ... is taken by
                    Possible hit: ... would take it
                    Note: First % is to avoid e.g. 'mis' prefix
                          Second % is to avoid e.g. 'ing' suffix
Master-pattern note: It is case insensitive with wildcards '*','@','#','$','%'
                     allowed. The purpose of this pattern is to
                     decide whether a search for next patterns will be
                     executed, it is applied on all lines i.e. the whole file.
                     There must be at least one hit in order to execute search
                     for next patterns.

Have a nice Kazuyaing.
For contacts: sanmayce@sanmayce.com
Sanmayce Svalqyatchx 'Kaze', 2010 May 24.

Allocated memory for DEcoded file in MB: 256
Kazuya: Can't open /? file.

D:\_KAZE_G.S._Corpus>
```


_KAZE_G.S._Corpus
Address D:_KAZE_G.S._Corpus
_KAZE_G.S._Corpus Properties
Type: File Folder
Location: D:\
Size: 6.56 GB (7,047,687,244 bytes)
Size on disk: 6.56 GB (7,049,113,600 bytes)
Contains: 660 Files, 5 Folders
5.90 GB
My Computer
Go to PROMPT
8:55 AM

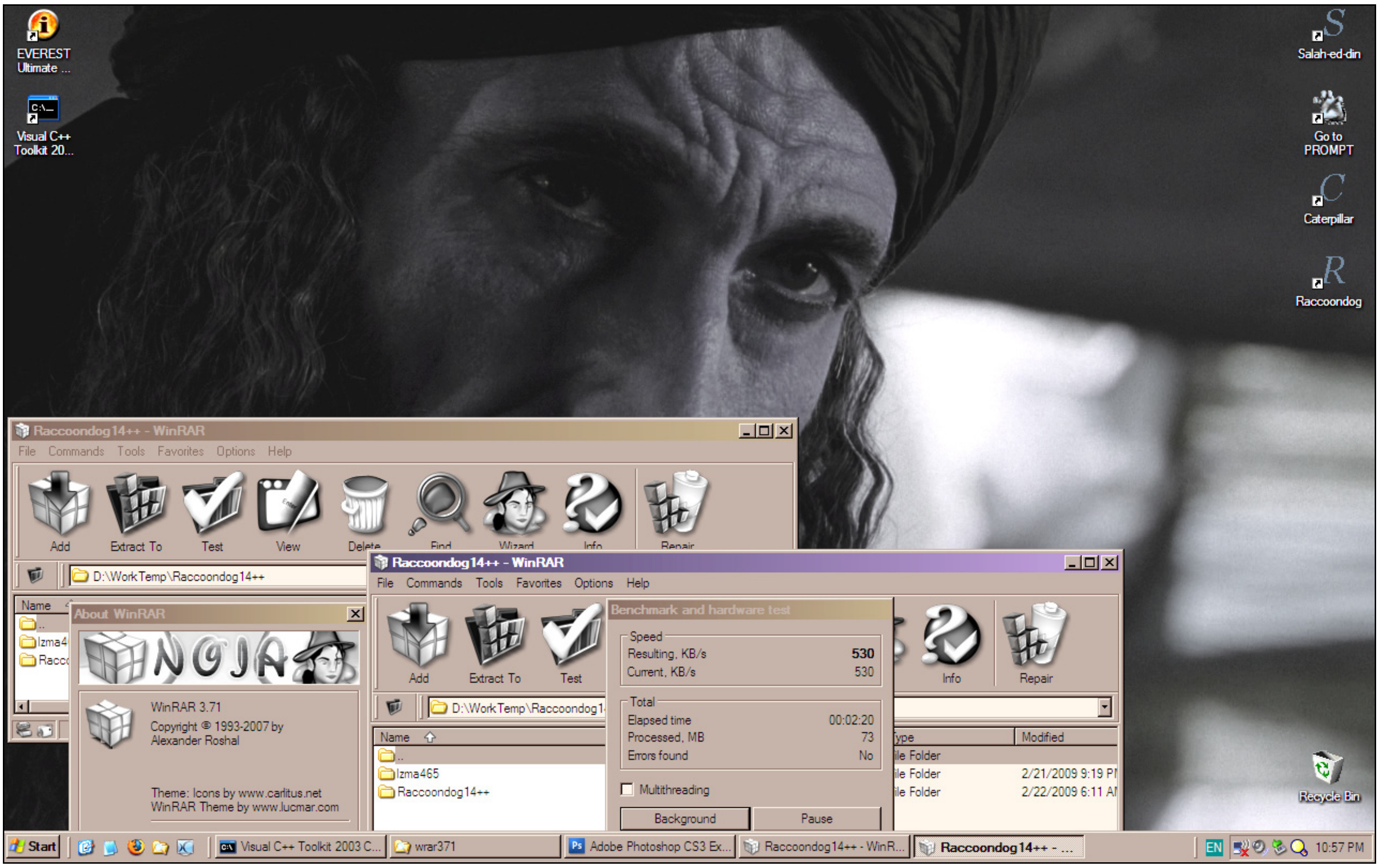

_KAZE_G.S._Corpus
Address D:_KAZE_G.S._Corpus
Open
Edit
Print
7-Zip
Convert to Adobe PDF
Convert to Adobe PDF and EMail
Bulk Rename Here
UltraEdit
Add to archive...
Add to "Caterpillar_HDD2RAM.rar"
Compress and email...
Compress to "Caterpillar_HDD2RAM.rar" and email
Load in Ultra Compare
Send To
Cut
Copy
Create Shortcut
Delete
Rename
Properties
Adobe Audition 3.0
Compressed (zipped) Folder
Desktop (create shortcut)
IsoBuster
Mail Recipient
My Documents
Skype
Sanmayce_Hitachi-298GB (H:)
Go to PROMPT
8:57 AM

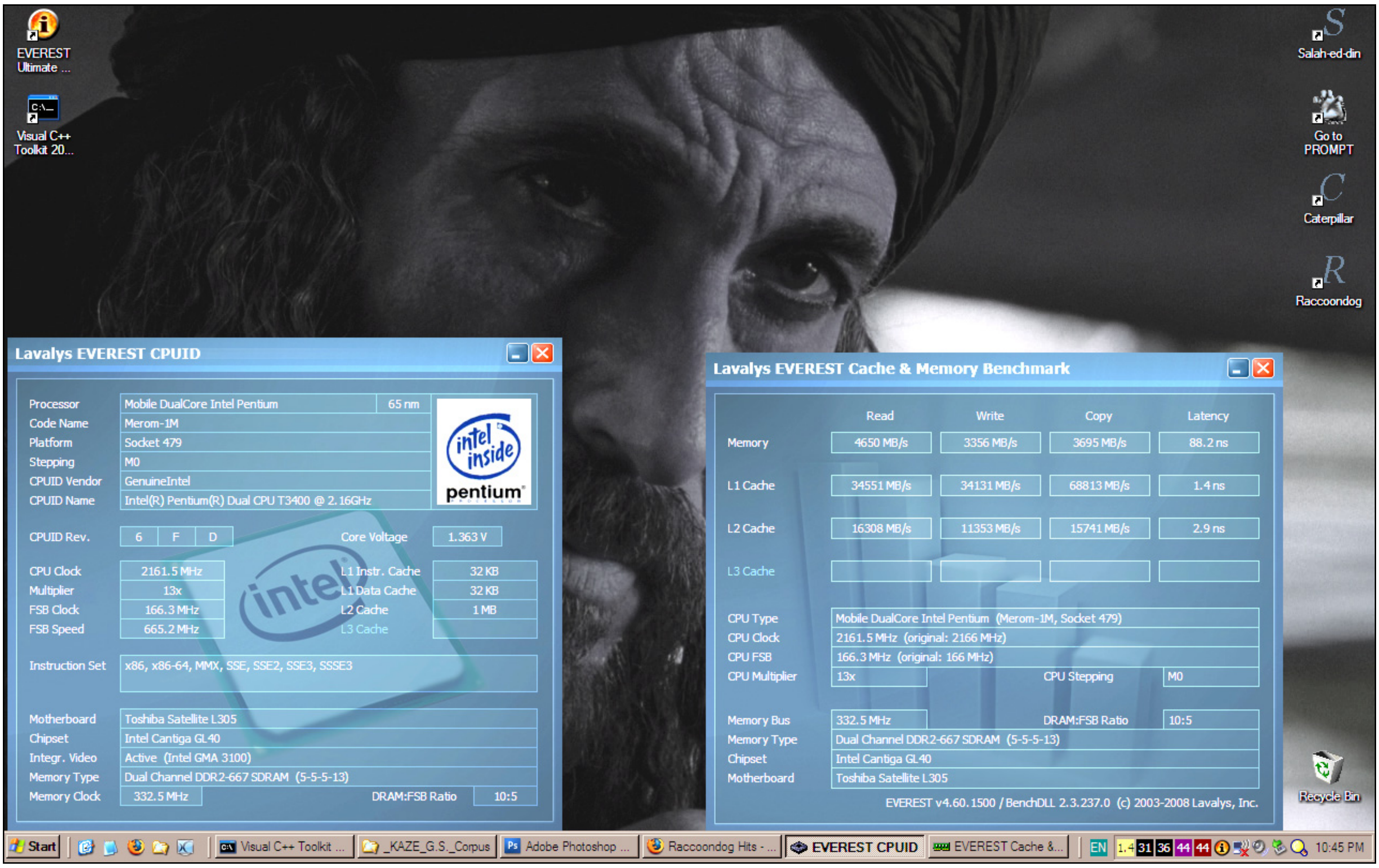


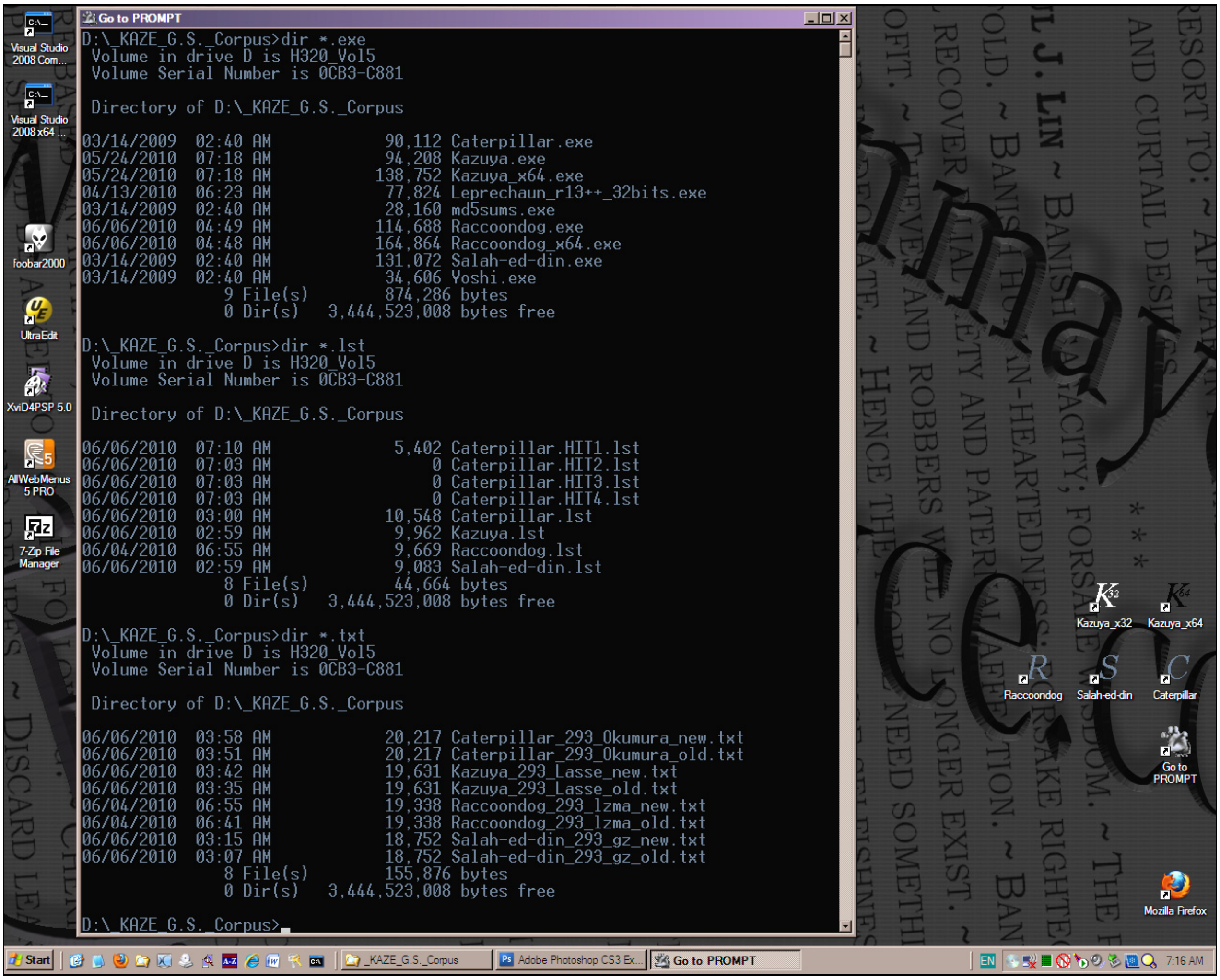
```
file:///D:/_KAZE_G.S._Corpus/_SearchIntoRafts/KAZE_G.S._Corpus_'.pdf'_Caterpillar.html
Book of Lao Zi: Dao De ... | Nifty console tools and L... | Kazuya Hits | Raccoondog Hits | Salah-ed-din Hits | Caterpillar Hits | Caterpillar Hits

01/25/2009  05:02 AM         1,154,717 Evil Under the Sun By Agatha Christie.pdf
01/25/2009  05:02 AM           479,173 Five Little Pigs By Agatha Christie.pdf
01/25/2009  05:02 AM           473,007 Halloween Party By Agatha Christie.pdf
01/25/2009  05:02 AM         1,286,369 Hercule Poirot's Christmas By Agatha Christie.pdf
01/25/2009  05:02 AM           292,411 Hickory Dickory Death By Agatha Christie.pdf
01/25/2009  05:02 AM           668,622 Labours Of Hercules By Agatha Christie.pdf
01/25/2009  05:02 AM         1,236,545 Lord Edgware Dies By Agatha Christie.pdf
01/25/2009  05:02 AM           540,449 Mrs Mcgintys Dead By Agatha Christie.pdf
01/25/2009  05:02 AM           421,856 Murder At The Vicarage By Agatha Christie.pdf
01/25/2009  05:02 AM         1,254,714 Murder in Mesopotamia By Agatha Christie.pdf
01/25/2009  05:02 AM           386,129 Murder Is Easy By Agatha Christie.pdf
01/25/2009  05:02 AM           345,254 Murder Of Roger Ackroyd By Agatha Christie.pdf
01/25/2009  05:02 AM           334,816 Murder on the Links By Agatha Christie.pdf
01/25/2009  05:02 AM           770,946 Murder On The Orient Express By Agatha Christie.pdf
01/25/2009  05:02 AM           540,272 Mystery Of The Blue Train By Agatha Christie.pdf
01/25/2009  05:02 AM           434,242 N Or M By Agatha Christie.pdf
01/25/2009  05:02 AM           561,502 Nemesis By Agatha Christie.pdf
01/25/2009  05:02 AM           399,098 One Two Buckle My Shoe By Agatha Christie.pdf
01/25/2009  05:02 AM           794,314 Parker Pyne Investigates By Agatha Christie.pdf
01/25/2009  05:02 AM           436,599 Partners In Crime By Agatha Christie.pdf
01/25/2009  05:02 AM           443,521 Passenger To Frankfurt By Agatha Christie.pdf
01/25/2009  05:02 AM           596,994 Peril At End House By Agatha Christie.pdf
01/25/2009  05:02 AM           425,371 Poirot's Early Cases By Agatha Christie.pdf
01/25/2009  05:02 AM           791,030 Sad Cypress By Agatha Christie.pdf
01/25/2009  05:02 AM           598,974 Sittaford Mystery By Agatha Christie.pdf
01/25/2009  05:02 AM           304,772 Sleeping Murder By Agatha Christie.pdf
01/25/2009  05:02 AM           510,546 Sparkling Cyanide By Agatha Christie.pdf
01/25/2009  05:02 AM           488,436 Surprise Surprise.pdf
01/25/2009  05:02 AM           539,352 Taken At The Flood By Agatha Christie.pdf
01/25/2009  05:02 AM           466,081 The Abc Murders By Agatha Christie.pdf
01/25/2009  05:02 AM           308,229 The Body In The Library By Agatha Christie.pdf
01/25/2009  05:02 AM           362,240 The Burden By Agatha Christie.pdf
01/25/2009  05:02 AM           647,357 The Casebook Of Hercule Poirot.pdf
01/25/2009  05:02 AM           343,894 The Circular Staircase.pdf
01/25/2009  05:02 AM           497,478 The Clocks By Agatha Christie.pdf
01/25/2009  05:02 AM           579,351 The Hollow By Agatha Christie.pdf
01/25/2009  05:02 AM           330,173 The Man In Lower Ten.pdf
01/25/2009  05:02 AM           595,817 The Man In The Brown Suit By Agatha Christie.pdf
01/25/2009  05:02 AM           344,868 The mirror cracked from side to side By Agatha Christie.pdf
01/25/2009  05:02 AM           269,020 The Moving Finger By Agatha Christie.pdf
01/25/2009  05:02 AM           251,750 The Mysterious Affair At Styles By Agatha Christie.pdf
01/25/2009  05:02 AM           298,070 The Mysterious Mr Quin By Agatha Christie.pdf
01/25/2009  05:02 AM           440,713 The Reggata Mystery By Agatha Christie.pdf
01/25/2009  05:02 AM           327,430 The Secret Adversary By Agatha Christie.pdf
01/25/2009  05:02 AM           541,063 The Secret Of Chimneys By Agatha Christie.pdf
01/25/2009  05:02 AM           493,592 The Seven Dials Mystery By Agatha Christie.pdf
01/25/2009  05:02 AM         1,225,944 They Came to Baghdad By Agatha Christie.pdf
01/25/2009  05:02 AM           289,044 They Do It With Mirrors By Agatha Christie.pdf
01/25/2009  05:02 AM           304,417 Third Girl By Agatha Christie.pdf
01/25/2009  05:02 AM           472,152 Three Act Tragedy By Agatha Christie.pdf
01/25/2009  05:02 AM         1,174,993 Three Blind Mice and Other Stories - Agatha Christie - 1948.pdf
01/25/2009  05:02 AM           582,226 Towards Zero By Agatha Christie.pdf
01/25/2009  05:02 AM           357,435 Why Didn't They Ask Evans By Agatha Christie.pdf
002,512  [*.pdf?][][][]  Caterpillar link to target file: D:\_KAZE_G.S._Corpus\_SearchIntoRafts\_RAFTbyRAFT_Description\RAFT0_og_on_s.dir
06/04/2010  04:49 AM        14,462,269 100 Questions and Answers About Alcoholism.pdf
06/04/2010  04:49 AM        12,167,723 210_Knots.pdf
06/04/2010  04:49 AM        52,613,250 Berkshire Encyclopedia Of World History Vol I - Abraham to Coal.pdf
06/04/2010  04:49 AM        43,800,823 Berkshire Encyclopedia Of World History Vol II - Cold War to Global Imperialism and Gender.pdf
06/04/2010  04:49 AM        51,772,828 Berkshire Encyclopedia Of World History Vol III - Global Migrations in Modern Time to Mysticism.pdf
```

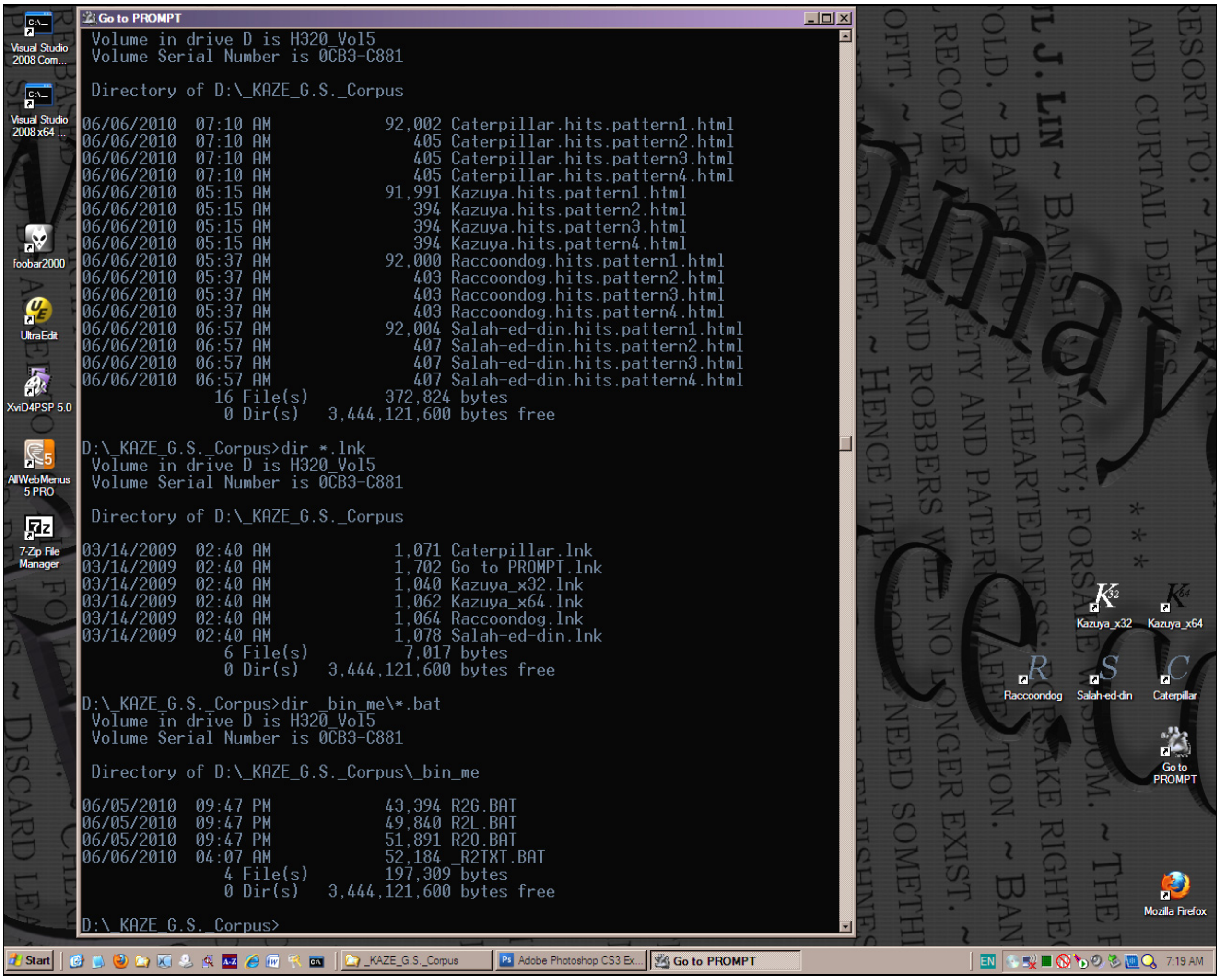


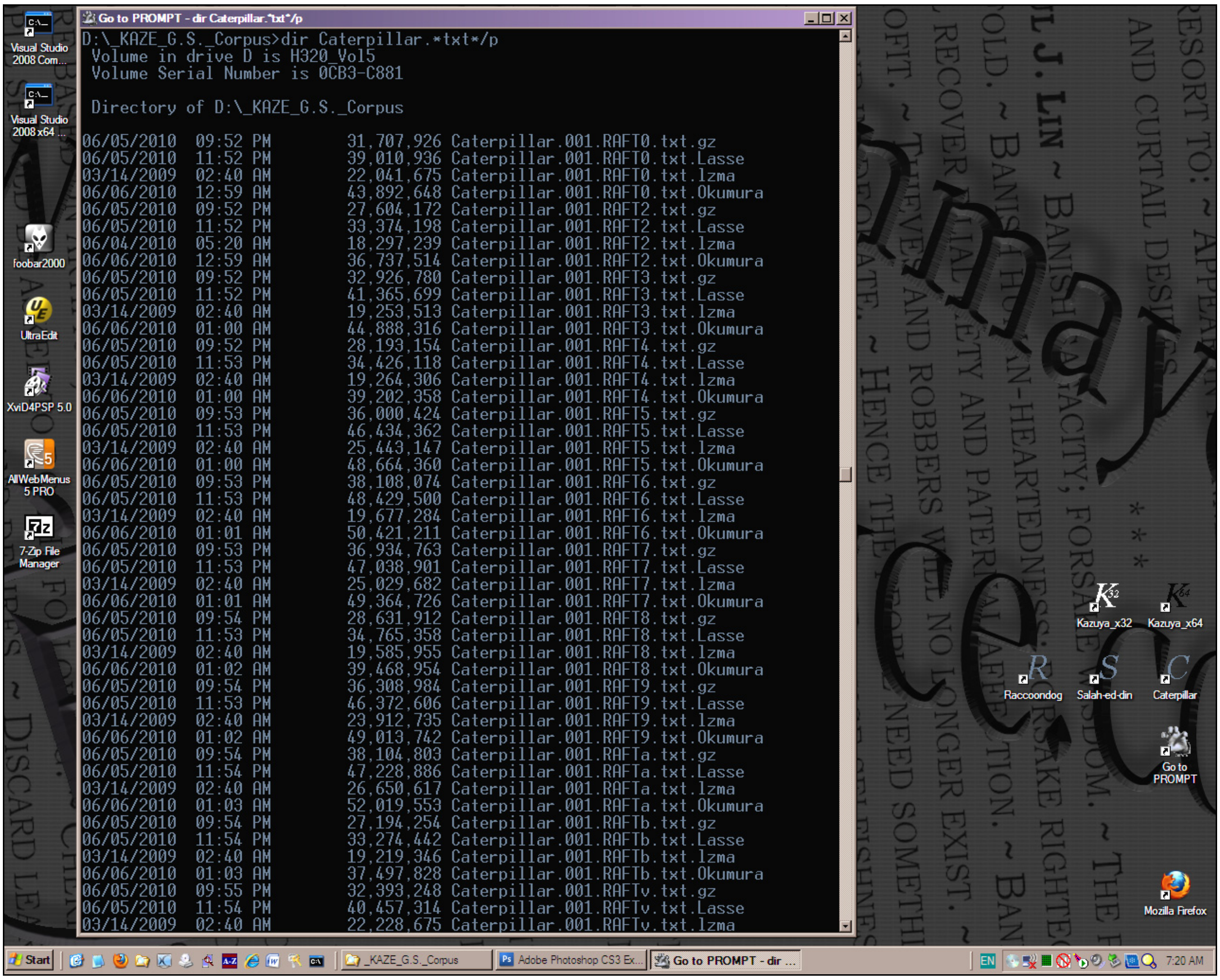
```
file:///D:/_KAZE_G.S._Corpus/_SearchIntoRafts/KAZE_G.S._Corpus_'.chm'_Caterpillar.html
Book of Lao Zi: Da... | Nifty console tools a... | Kazuya Hits | Raccoondog Hits | Salah-ed-din Hits | Caterpillar Hits | Caterpillar Hits | Caterpillar Hits

01/25/2009  05:02 AM         7,815,498 A Student's Guide to Textual Criticism of the Bible.chm
01/25/2009  05:02 AM         1,817,974 Death and the Afterlife - A Cultural Encyclopedia.chm
01/25/2009  05:02 AM         1,315,722 Textual Commentary on the Greek New Testament.chm
000,003  [*.chm?][][][]  Caterpillar link to target file: D:\_KAZE_G.S._Corpus\_SearchIntoRafts\_RAFTbyRAFT_Description\RAFT0_og_
06/04/2010  04:49 AM         1,323,773 FatwaBase - v4.99.chm
06/04/2010  04:49 AM         2,317,291 Hisnul Muslim.chm
06/04/2010  04:49 AM         5,963,547 ibnkathir_english (TAFSIR).chm
06/04/2010  04:49 AM         1,512,921 Islam Against Terrorism - v1.33.chm
06/04/2010  04:49 AM         3,644,892 prophetsprayer.chm
06/04/2010  04:49 AM        31,516,757 quransahih.chm
06/04/2010  04:49 AM         7,941,208 Sahih Bukhari, Muslim, Muwatta _ Abu Dawood.chm
06/04/2010  04:49 AM         8,418,464 Sahih Bukhari.chm
06/04/2010  04:49 AM         2,501,422 sahih_bukhari_01.chm
06/04/2010  04:49 AM         2,244,116 sahih_muslim_01.chm
06/04/2010  04:49 AM        32,751,405 The Noble Quran.chm
06/04/2010  04:49 AM         3,644,892 The Prophet's Prayer - v3.20.chm
06/04/2010  04:49 AM        87,560,405 Complete Works of Osho.chm
000,013  [*.chm?][][][]  Caterpillar link to target file: D:\_KAZE_G.S._Corpus\_SearchIntoRafts\_RAFTbyRAFT_Description\RAFT2_og_
12/02/2006  10:18 PM        26,168,854 3d Max User Reference.chm
12/02/2006  10:18 PM        37,108,777 60 Common Web Design Mistakes and How to Avoid Them.chm
12/02/2006  10:18 PM         1,526,125 70.214.examcram2.chm
12/02/2006  10:18 PM         2,092,162 A First Look at ADO.NET and System Xml 2.0.chm
12/02/2006  10:18 PM         6,323,265 A Guide to Constructing GUIs.chm
12/02/2006  10:18 PM        10,160,148 A Roadmap for Building a Linux File and Print Server.chm
12/02/2006  10:18 PM         7,129,066 A+ Certification Training Kit - Second Edition.chm
12/02/2006  10:18 PM         3,904,279 Absolute Beginner's Guide to Launching an eBay Business.chm
12/02/2006  10:18 PM         1,071,122 Absolute OpenBSD - UNIX for the Practical Paranoid.chm
12/02/2006  10:18 PM         1,577,580 ACADP9.CHM
12/02/2006  10:18 PM         3,516,968 ACMAIN9.CHM
12/02/2006  10:18 PM         2,054,031 ACTIONSCRIPT FOR FLASH MX - THE DEFINITIVE GUIDE, 2ND EDITION.CHM
12/02/2006  10:18 PM         2,092,162 Addison.Wesley.A.First.Look.At.ADO.Dot.NET.And.System.Xml.v.2.0.eBook-LiB.chm
12/02/2006  10:18 PM         4,385,293 Addison.Wesley.Algorithms.In.Java.3rd.Ed.Part5.Graph.Algorithms.eBook-LiB.chm
12/02/2006  10:18 PM         2,057,225 Addison.Wesley.Parallel.And.Distributed.Programming.Using.Cpp.eBook-LiB.chm
12/02/2006  10:18 PM         1,565,802 Addison.Wesley.The.Art.Of.Unix.Programming.eBook-LiB.chm
12/02/2006  10:18 PM         7,493,194 Addison-Wesley - A Programmers Guide to .NET.chm
12/02/2006  10:18 PM        14,111,320 Administering and Securing the Apache Server.chm
12/02/2006  10:18 PM         1,384,762 ADO210.CHM
12/02/2006  10:18 PM         4,301,415 Advanced MS Visual Basic 6.0 Second Edition.chm
12/02/2006  10:18 PM         8,283,383 Advantage Database Server - The Official Guide.chm
12/02/2006  10:18 PM        11,260,817 Algorithms for Compiler Design.chm
12/02/2006  10:18 PM         4,385,293 Algorithms in Java, Third Edition, Part 5 - Graph Algorithms.chm
12/02/2006  10:18 PM        12,085,560 A-LIST.Publishing.Modern.Cryptography.eBook-LiB.chm
12/02/2006  10:18 PM        30,245,817 Anti-Hacker Tool Kit.chm
12/02/2006  10:18 PM         4,471,967 Applied Software Engineering Using Apache Jakarta Commons.chm
12/02/2006  10:18 PM         1,932,072 Applying.Enterprise.JavaBeans.2nd.Edition.chm
12/02/2006  10:18 PM        15,773,791 Architecting Portal Solutions.chm
12/02/2006  10:18 PM        29,025,837 ASP.NET Programming with Visual C# .NET 2003 Step by Step.chm
12/02/2006  10:18 PM         8,977,185 Assembly Language Step-by-Step - Programming with DOS and Linux.chm
12/02/2006  10:18 PM         1,015,798 ax_enu.chm
12/02/2006  10:18 PM         4,802,614 Beowulf.Cluster.Computing.With.Linux.Second.Edition.eBook-Li.chm
12/02/2006  10:18 PM         3,165,168 BeyondDreamweaver.chm
12/02/2006  10:18 PM         8,924,654 Building Dynamic Websites with Macromedia Studio MX 2004.chm
```


EVEREST Ultimate ...
Visual C++ Toolkit 20...
Salah-ed-din
Go to PROMPT
Caterpillar
Raccoondog
Recycle Bin
Raccoondog14++ - WinRAR
File Commands Tools Favorites Options Help
Add Extract To Test View Delete Find Wizard Info Repair
D:\WorkTemp\Raccoondog14++
About WinRAR
WinRAR 3.71
Copyright © 1993-2007 by Alexander Roshal
Theme: Icons by www.carlitus.net
WinRAR Theme by www.lucmar.com
Benchmark and hardware test
Speed
Resulting, KB/s 530
Current, KB/s 530
Total
Elapsed time 00:02:20
Processed, MB 73
Errors found No
Multithreading
Background
Pause
Name
lzma465
Raccoondog14++
Modified
2/21/2009 9:19 PM
2/22/2009 6:11 AM
Start
Visual C++ Toolkit 2003 C...
wrar371
Adobe Photoshop CS3 Ex...
Raccoondog14++ - WinR...
Raccoondog14++ - ...
10:57 PM

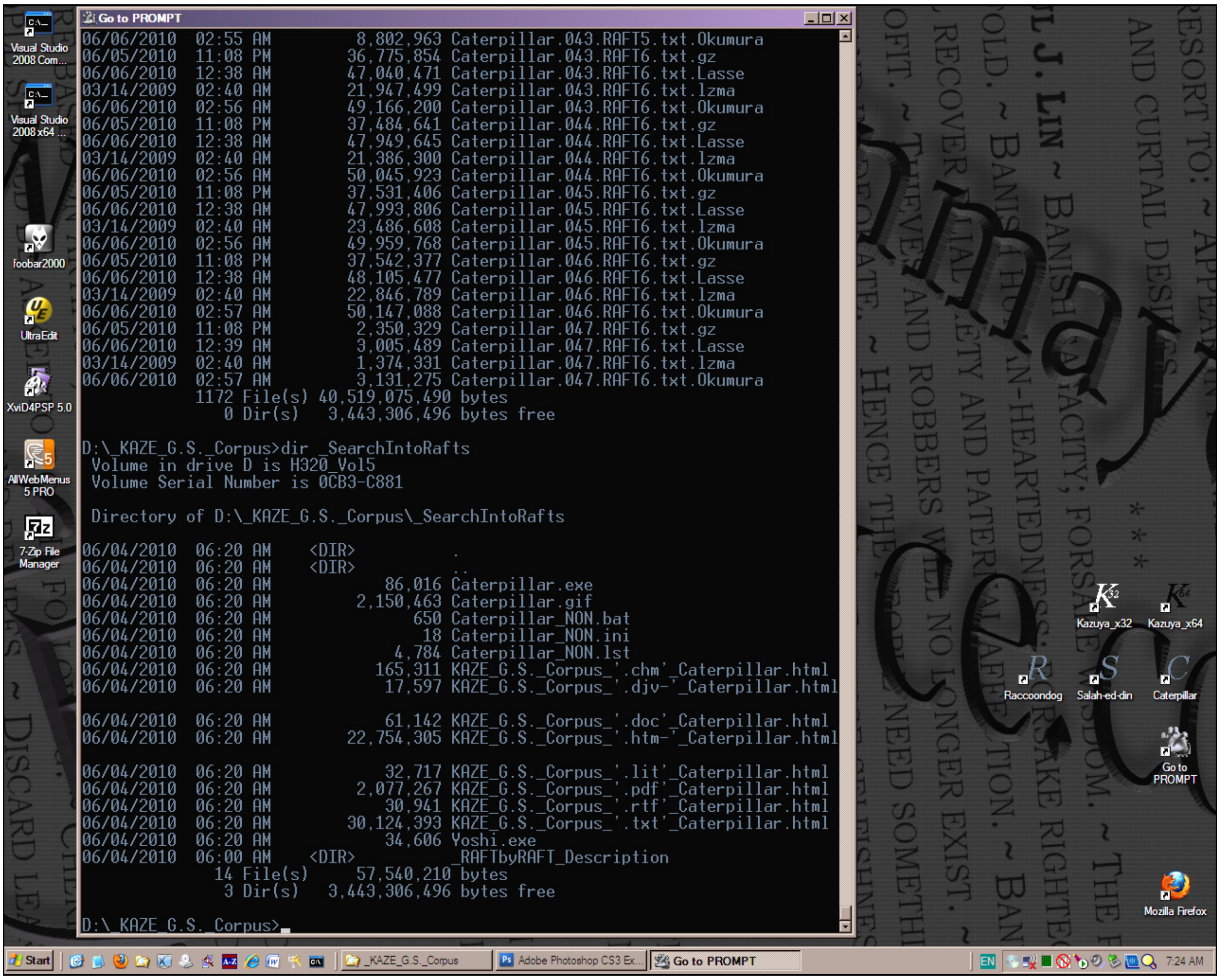

Lavalys EVEREST CPUID
Processor Mobile DualCore Intel Pentium 65 nm
Code Name Merom-1M
Platform Socket 479
Stepping M0
CPUID Vendor GenuineIntel
CPUID Name Intel(R) Pentium(R) Dual CPU T3400 @ 2.16GHz
CPUID Rev. 6 F D
Core Voltage 1.363 V
CPU Clock 2161.5 MHz
Multiplier 13x
FSB Clock 166.3 MHz
FSB Speed 665.2 MHz
L1 Instr. Cache 32 KB
L1 Data Cache 32 KB
L2 Cache 1 MB
L3 Cache
Instruction Set x86, x86-64, MMX, SSE, SSE2, SSE3, SSSE3
Motherboard Toshiba Satellite L305
Chipset Intel Cantiga GL40
Integr. Video Active (Intel GMA 3100)
Memory Type Dual Channel DDR2-667 SDRAM (5-5-5-13)
Memory Clock 332.5 MHz
DRAM:FSB Ratio 10:5
Lavalys EVEREST Cache & Memory Benchmark
Read Write Copy Latency
Memory 4650 MB/s 3356 MB/s 3695 MB/s 88.2 ns
L1 Cache 34551 MB/s 34131 MB/s 68813 MB/s 1.4 ns
L2 Cache 16308 MB/s 11353 MB/s 15741 MB/s 2.9 ns
L3 Cache
CPU Type Mobile DualCore Intel Pentium (Merom-1M, Socket 479)
CPU Clock 2161.5 MHz (original: 2166 MHz)
CPU FSB 166.3 MHz (original: 166 MHz)
CPU Multiplier 13x
CPU Stepping M0
Memory Bus 332.5 MHz
DRAM:FSB Ratio 10:5
Memory Type Dual Channel DDR2-667 SDRAM (5-5-5-13)
Chipset Intel Cantiga GL40
Motherboard Toshiba Satellite L305
EVEREST v4.60.1500 / BenchDLL 2.3.237.0 (c) 2003-2008 Lavalys, Inc.
Start
Visual C++ Toolkit ...
_KAZE_G.S._Corpus
Adobe Photoshop ...
Raccoondog Hits - ...
EVEREST CPUID
EVEREST Cache &...
10:45 PM


Go to PROMPT
D:_KAZE_G.S._Corpus>dir *.exe
Volume in drive D is H320_Vol5
Volume Serial Number is 0CB3-C881
Directory of D:_KAZE_G.S._Corpus
03/14/2009 02:40 AM 90,112 Caterpillar.exe
05/24/2010 07:18 AM 94,208 Kazuya.exe
05/24/2010 07:18 AM 138,752 Kazuya_x64.exe
04/13/2010 06:23 AM 77,824 Leprechaun_r13++_32bits.exe
03/14/2009 02:40 AM 28,160 md5sums.exe
06/06/2010 04:49 AM 114,688 Raccoondog.exe
06/06/2010 04:48 AM 164,864 Raccoondog_x64.exe
03/14/2009 02:40 AM 131,072 Salah-ed-din.exe
03/14/2009 02:40 AM 34,606 Yoshi.exe
9 File(s) 874,286 bytes
0 Dir(s) 3,444,523,008 bytes free
D:_KAZE_G.S._Corpus>dir *.lst
Volume in drive D is H320_Vol5
Volume Serial Number is 0CB3-C881
Directory of D:_KAZE_G.S._Corpus
06/06/2010 07:10 AM 5,402 Caterpillar.HIT1.lst
06/06/2010 07:03 AM 0 Caterpillar.HIT2.lst
06/06/2010 07:03 AM 0 Caterpillar.HIT3.lst
06/06/2010 07:03 AM 0 Caterpillar.HIT4.lst
06/06/2010 03:00 AM 10,548 Caterpillar.lst
06/06/2010 02:59 AM 9,962 Kazuya.lst
06/04/2010 06:55 AM 9,669 Raccoondog.lst
06/06/2010 02:59 AM 9,083 Salah-ed-din.lst
8 File(s) 44,664 bytes
0 Dir(s) 3,444,523,008 bytes free
D:_KAZE_G.S._Corpus>dir *.txt
Volume in drive D is H320_Vol5
Volume Serial Number is 0CB3-C881
Directory of D:_KAZE_G.S._Corpus
06/06/2010 03:58 AM 20,217 Caterpillar_293_Okumura_new.txt
06/06/2010 03:51 AM 20,217 Caterpillar_293_Okumura_old.txt
06/06/2010 03:42 AM 19,631 Kazuya_293_Lasse_new.txt
06/06/2010 03:35 AM 19,631 Kazuya_293_Lasse_old.txt
06/04/2010 06:55 AM 19,338 Raccoondog_293_lzma_new.txt
06/04/2010 06:41 AM 19,338 Raccoondog_293_lzma_old.txt
06/06/2010 03:15 AM 18,752 Salah-ed-din_293_gz_new.txt
06/06/2010 03:07 AM 18,752 Salah-ed-din_293_gz_old.txt
8 File(s) 155,876 bytes
0 Dir(s) 3,444,523,008 bytes free
D:_KAZE_G.S._Corpus>
Start
_KAZE_G.S._Corpus
Adobe Photoshop CS3 Ex...
Go to PROMPT
7:16 AM

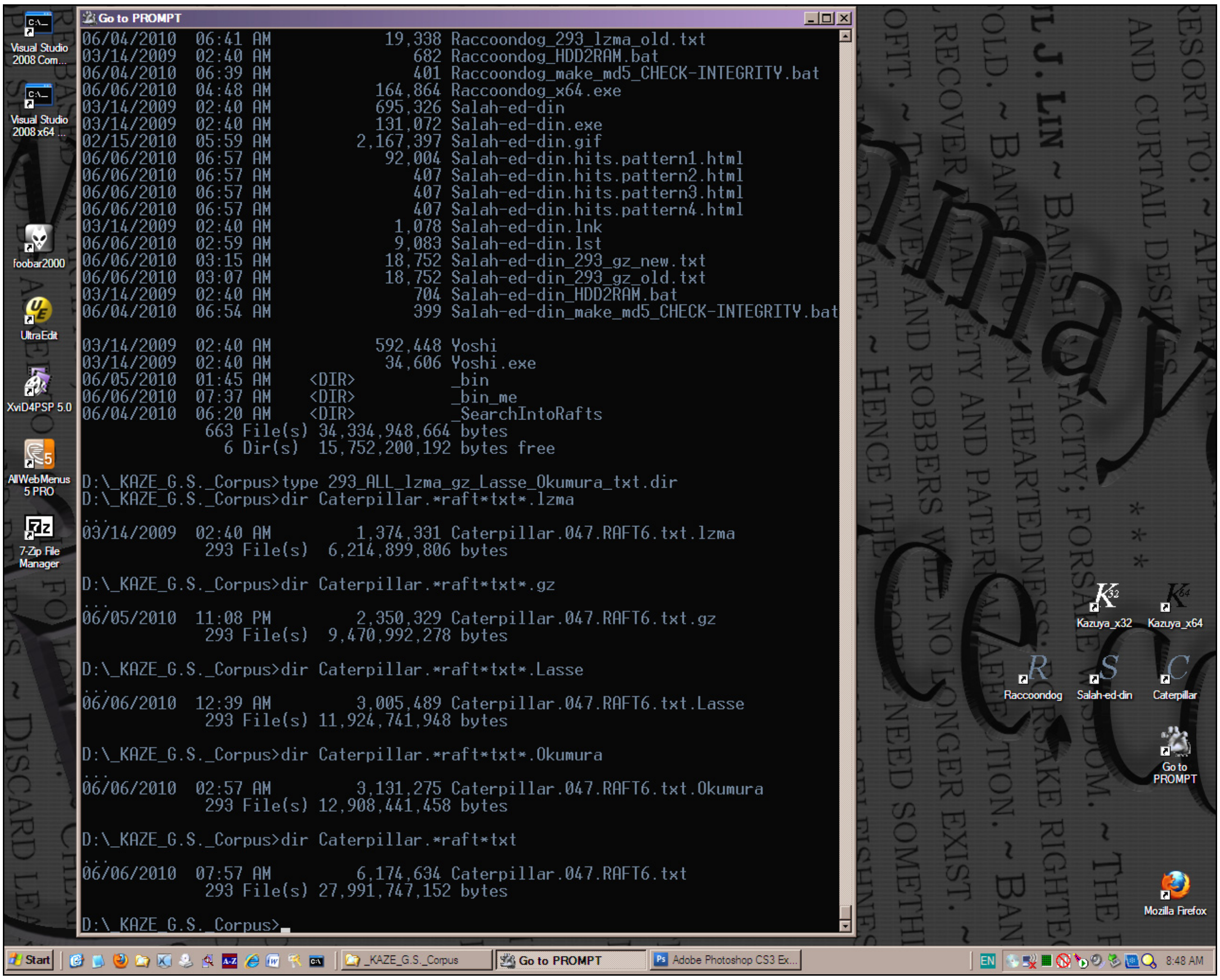

Go to PROMPT
Volume in drive D is H320_Vol5
Volume Serial Number is 0CB3-C881
Directory of D:_KAZE_G.S._Corpus
06/06/2010 07:10 AM 92,002 Caterpillar.hits.pattern1.html
06/06/2010 07:10 AM 405 Caterpillar.hits.pattern2.html
06/06/2010 07:10 AM 405 Caterpillar.hits.pattern3.html
06/06/2010 07:10 AM 405 Caterpillar.hits.pattern4.html
06/06/2010 05:15 AM 91,991 Kazuya.hits.pattern1.html
06/06/2010 05:15 AM 394 Kazuya.hits.pattern2.html
06/06/2010 05:15 AM 394 Kazuya.hits.pattern3.html
06/06/2010 05:15 AM 394 Kazuya.hits.pattern4.html
06/06/2010 05:37 AM 92,000 Raccoondog.hits.pattern1.html
06/06/2010 05:37 AM 403 Raccoondog.hits.pattern2.html
06/06/2010 05:37 AM 403 Raccoondog.hits.pattern3.html
06/06/2010 05:37 AM 403 Raccoondog.hits.pattern4.html
06/06/2010 06:57 AM 92,004 Salah-ed-din.hits.pattern1.html
06/06/2010 06:57 AM 407 Salah-ed-din.hits.pattern2.html
06/06/2010 06:57 AM 407 Salah-ed-din.hits.pattern3.html
06/06/2010 06:57 AM 407 Salah-ed-din.hits.pattern4.html
16 File(s) 372,824 bytes
0 Dir(s) 3,444,121,600 bytes free
D:_KAZE_G.S._Corpus>dir *.lnk
Volume in drive D is H320_Vol5
Volume Serial Number is 0CB3-C881
Directory of D:_KAZE_G.S._Corpus
03/14/2009 02:40 AM 1,071 Caterpillar.lnk
03/14/2009 02:40 AM 1,702 Go to PROMPT.lnk
03/14/2009 02:40 AM 1,040 Kazuya_x32.lnk
03/14/2009 02:40 AM 1,062 Kazuya_x64.lnk
03/14/2009 02:40 AM 1,064 Raccoondog.lnk
03/14/2009 02:40 AM 1,078 Salah-ed-din.lnk
6 File(s) 7,017 bytes
0 Dir(s) 3,444,121,600 bytes free
D:_KAZE_G.S._Corpus>dir _bin_me*.bat
Volume in drive D is H320_Vol5
Volume Serial Number is 0CB3-C881
Directory of D:_KAZE_G.S._Corpus_bin_me
06/05/2010 09:47 PM 43,394 R2G.BAT
06/05/2010 09:47 PM 49,840 R2L.BAT
06/05/2010 09:47 PM 51,891 R2O.BAT
06/06/2010 04:07 AM 52,184 _R2TXT.BAT
4 File(s) 197,309 bytes
0 Dir(s) 3,444,121,600 bytes free
D:_KAZE_G.S._Corpus>
Start
_KAZE_G.S._Corpus
Adobe Photoshop CS3 Ex...
Go to PROMPT
7:19 AM

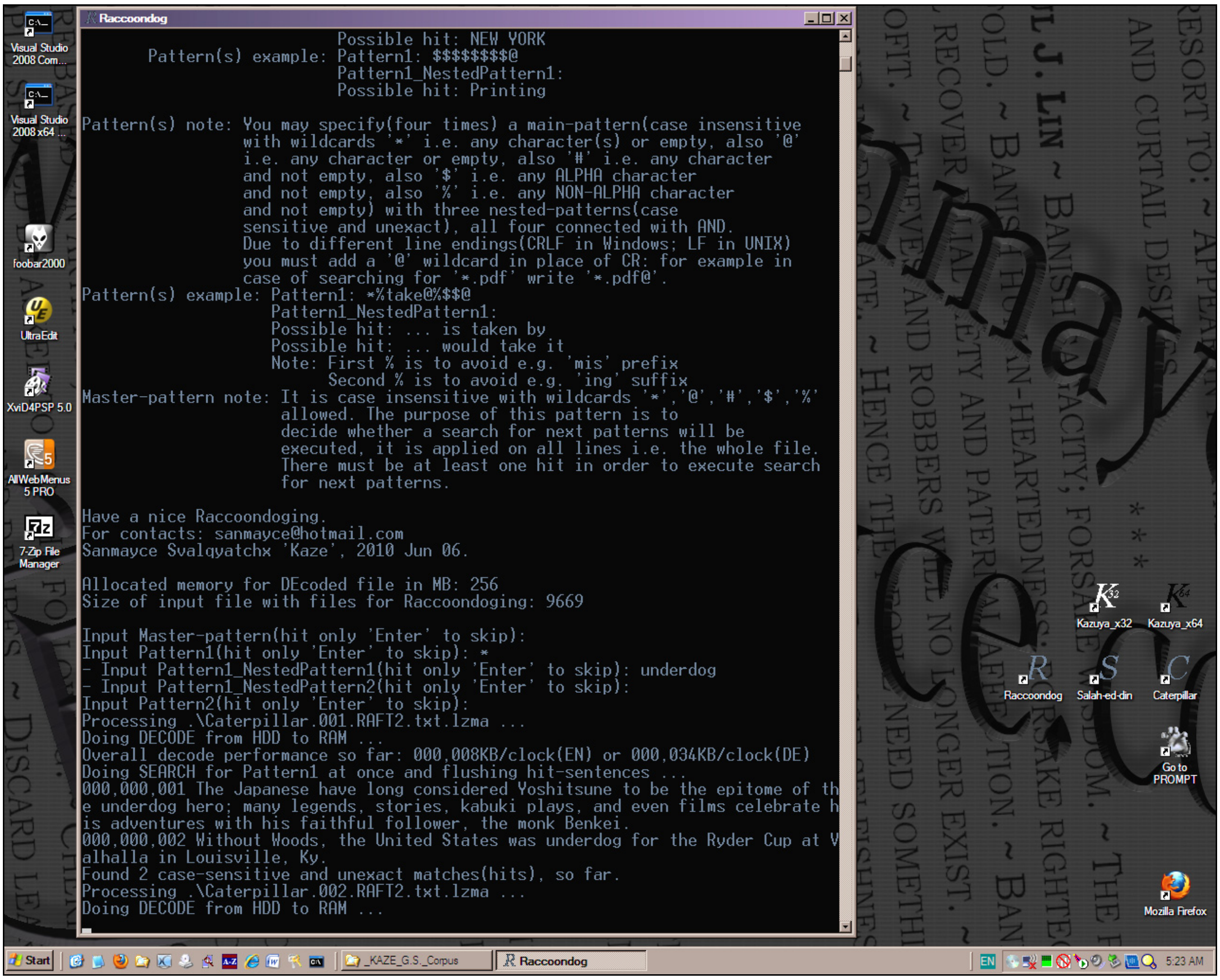


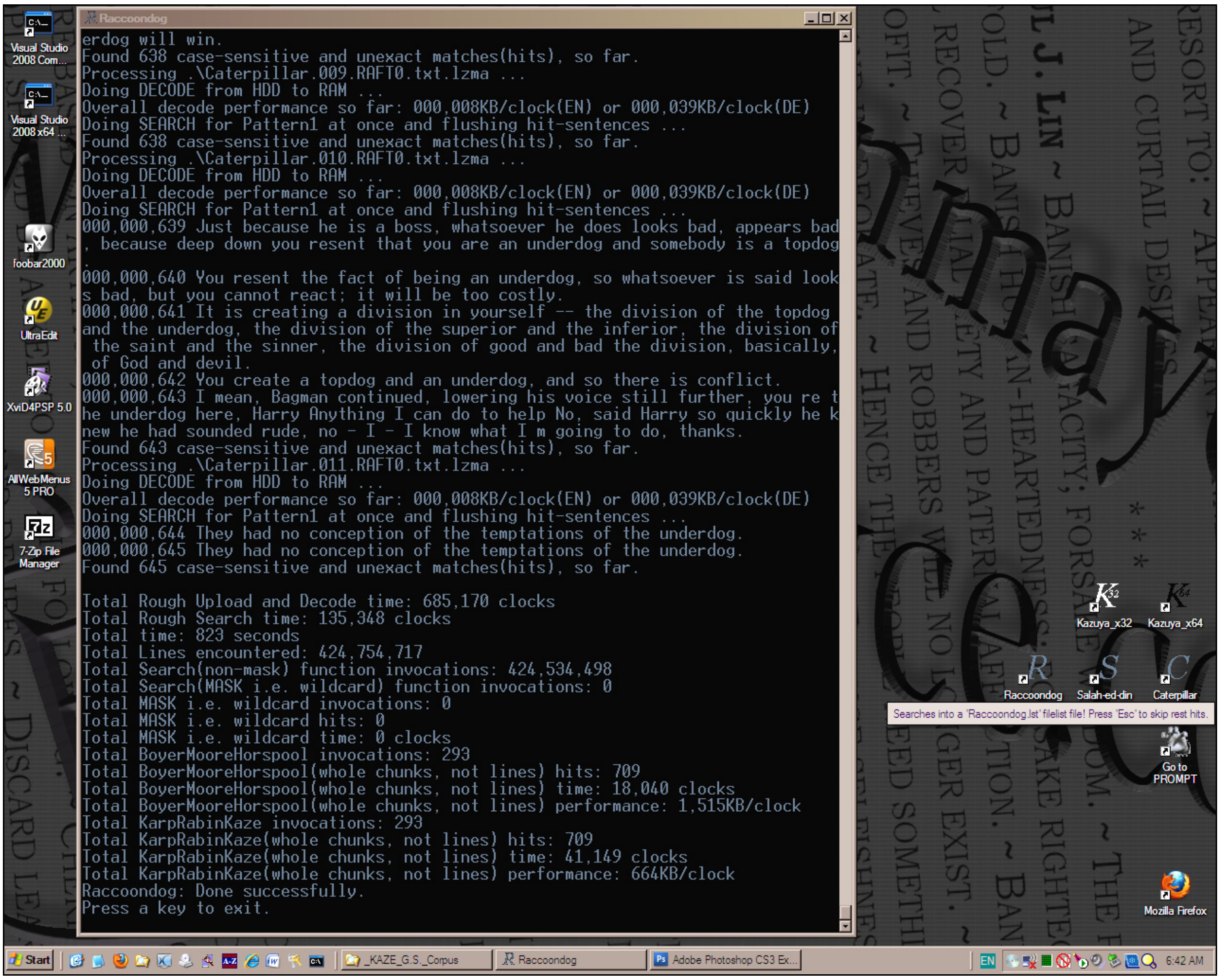
```
Go to PROMPT - dir Caterpillar.*txt*/p
D:\_KAZE_G.S._Corpus>dir Caterpillar.*txt*/p
 Volume in drive D is H320_Vol5
 Volume Serial Number is 0CB3-C881

 Directory of D:\_KAZE_G.S._Corpus

06/05/2010  09:52 PM        31,707,926 Caterpillar.001.RAFT0.txt.gz
06/05/2010  11:52 PM        39,010,936 Caterpillar.001.RAFT0.txt.Lasse
03/14/2009  02:40 AM        22,041,675 Caterpillar.001.RAFT0.txt.lzma
06/06/2010  12:59 AM        43,892,648 Caterpillar.001.RAFT0.txt.Okumura
06/05/2010  09:52 PM        27,604,172 Caterpillar.001.RAFT2.txt.gz
06/05/2010  11:52 PM        33,374,198 Caterpillar.001.RAFT2.txt.Lasse
06/04/2010  05:20 AM        18,297,239 Caterpillar.001.RAFT2.txt.lzma
06/06/2010  12:59 AM        36,737,514 Caterpillar.001.RAFT2.txt.Okumura
06/05/2010  09:52 PM        32,926,780 Caterpillar.001.RAFT3.txt.gz
06/05/2010  11:52 PM        41,365,699 Caterpillar.001.RAFT3.txt.Lasse
03/14/2009  02:40 AM        19,253,513 Caterpillar.001.RAFT3.txt.lzma
06/06/2010  01:00 AM        44,888,316 Caterpillar.001.RAFT3.txt.Okumura
06/05/2010  09:52 PM        28,193,154 Caterpillar.001.RAFT4.txt.gz
06/05/2010  11:53 PM        34,426,118 Caterpillar.001.RAFT4.txt.Lasse
03/14/2009  02:40 AM        19,264,306 Caterpillar.001.RAFT4.txt.lzma
06/06/2010  01:00 AM        39,202,358 Caterpillar.001.RAFT4.txt.Okumura
06/05/2010  09:53 PM        36,000,424 Caterpillar.001.RAFT5.txt.gz
06/05/2010  11:53 PM        46,434,362 Caterpillar.001.RAFT5.txt.Lasse
03/14/2009  02:40 AM        25,443,147 Caterpillar.001.RAFT5.txt.lzma
06/06/2010  01:00 AM        48,664,360 Caterpillar.001.RAFT5.txt.Okumura
06/05/2010  09:53 PM        38,108,074 Caterpillar.001.RAFT6.txt.gz
06/05/2010  11:53 PM        48,429,500 Caterpillar.001.RAFT6.txt.Lasse
03/14/2009  02:40 AM        19,677,284 Caterpillar.001.RAFT6.txt.lzma
06/06/2010  01:01 AM        50,421,211 Caterpillar.001.RAFT6.txt.Okumura
06/05/2010  09:53 PM        36,934,763 Caterpillar.001.RAFT7.txt.gz
06/05/2010  11:53 PM        47,038,901 Caterpillar.001.RAFT7.txt.Lasse
03/14/2009  02:40 AM        25,029,682 Caterpillar.001.RAFT7.txt.lzma
06/06/2010  01:01 AM        49,364,726 Caterpillar.001.RAFT7.txt.Okumura
06/05/2010  09:54 PM        28,631,912 Caterpillar.001.RAFT8.txt.gz
06/05/2010  11:53 PM        34,765,358 Caterpillar.001.RAFT8.txt.Lasse
03/14/2009  02:40 AM        19,585,955 Caterpillar.001.RAFT8.txt.lzma
06/06/2010  01:02 AM        39,468,954 Caterpillar.001.RAFT8.txt.Okumura
06/05/2010  09:54 PM        36,308,984 Caterpillar.001.RAFT9.txt.gz
06/05/2010  11:53 PM        46,372,606 Caterpillar.001.RAFT9.txt.Lasse
03/14/2009  02:40 AM        23,912,735 Caterpillar.001.RAFT9.txt.lzma
06/06/2010  01:02 AM        49,013,742 Caterpillar.001.RAFT9.txt.Okumura
06/05/2010  09:54 PM        38,104,803 Caterpillar.001.RAFTa.txt.gz
06/05/2010  11:54 PM        47,228,886 Caterpillar.001.RAFTa.txt.Lasse
03/14/2009  02:40 AM        26,650,617 Caterpillar.001.RAFTa.txt.lzma
06/06/2010  01:03 AM        52,019,553 Caterpillar.001.RAFTa.txt.Okumura
06/05/2010  09:54 PM        27,194,254 Caterpillar.001.RAFTb.txt.gz
06/05/2010  11:54 PM        33,274,442 Caterpillar.001.RAFTb.txt.Lasse
03/14/2009  02:40 AM        19,219,346 Caterpillar.001.RAFTb.txt.lzma
06/06/2010  01:03 AM        37,497,828 Caterpillar.001.RAFTb.txt.Okumura
06/05/2010  09:55 PM        32,393,248 Caterpillar.001.RAFTv.txt.gz
06/05/2010  11:54 PM        40,457,314 Caterpillar.001.RAFTv.txt.Lasse
03/14/2009  02:40 AM        22,228,675 Caterpillar.001.RAFTv.txt.lzma
```

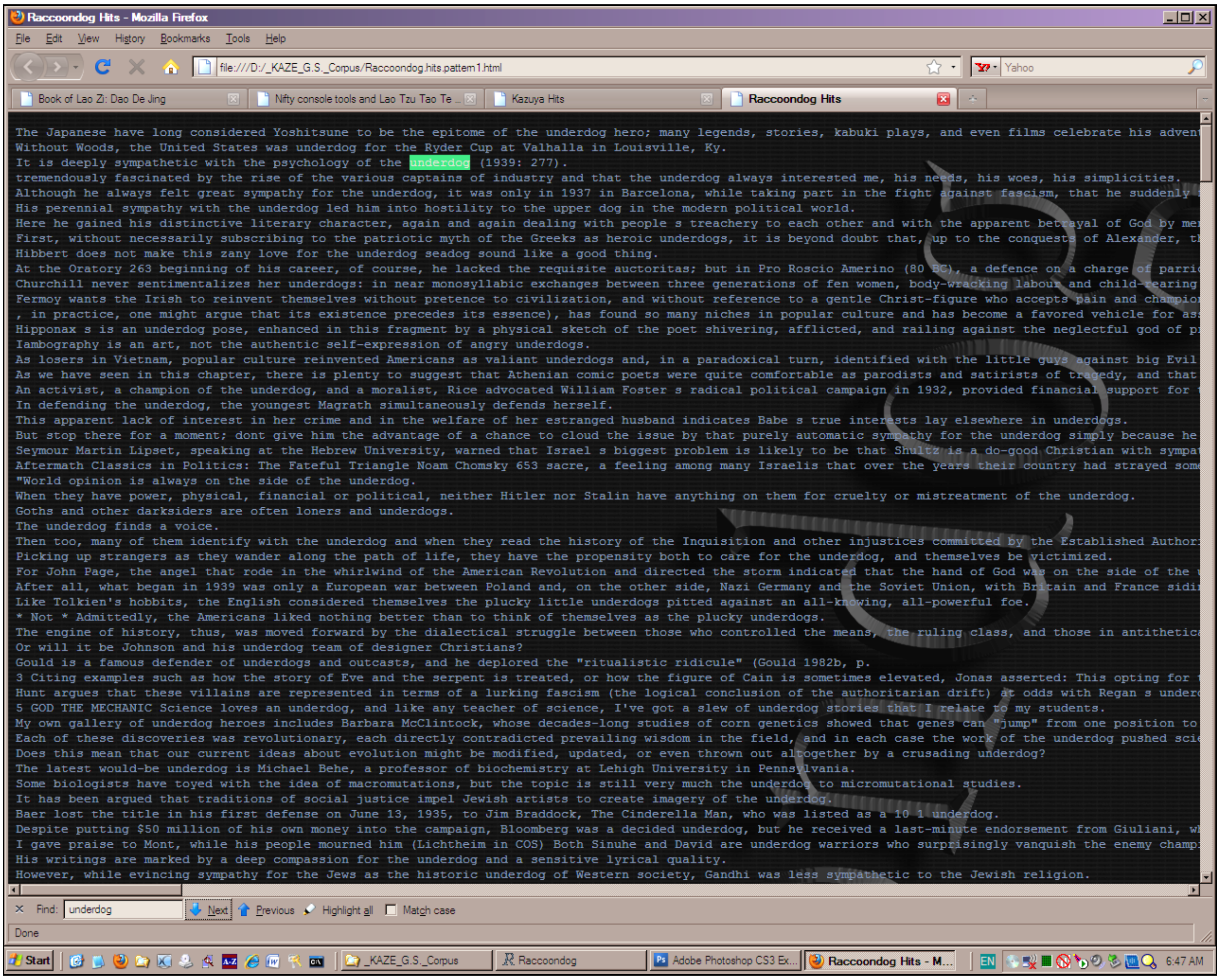


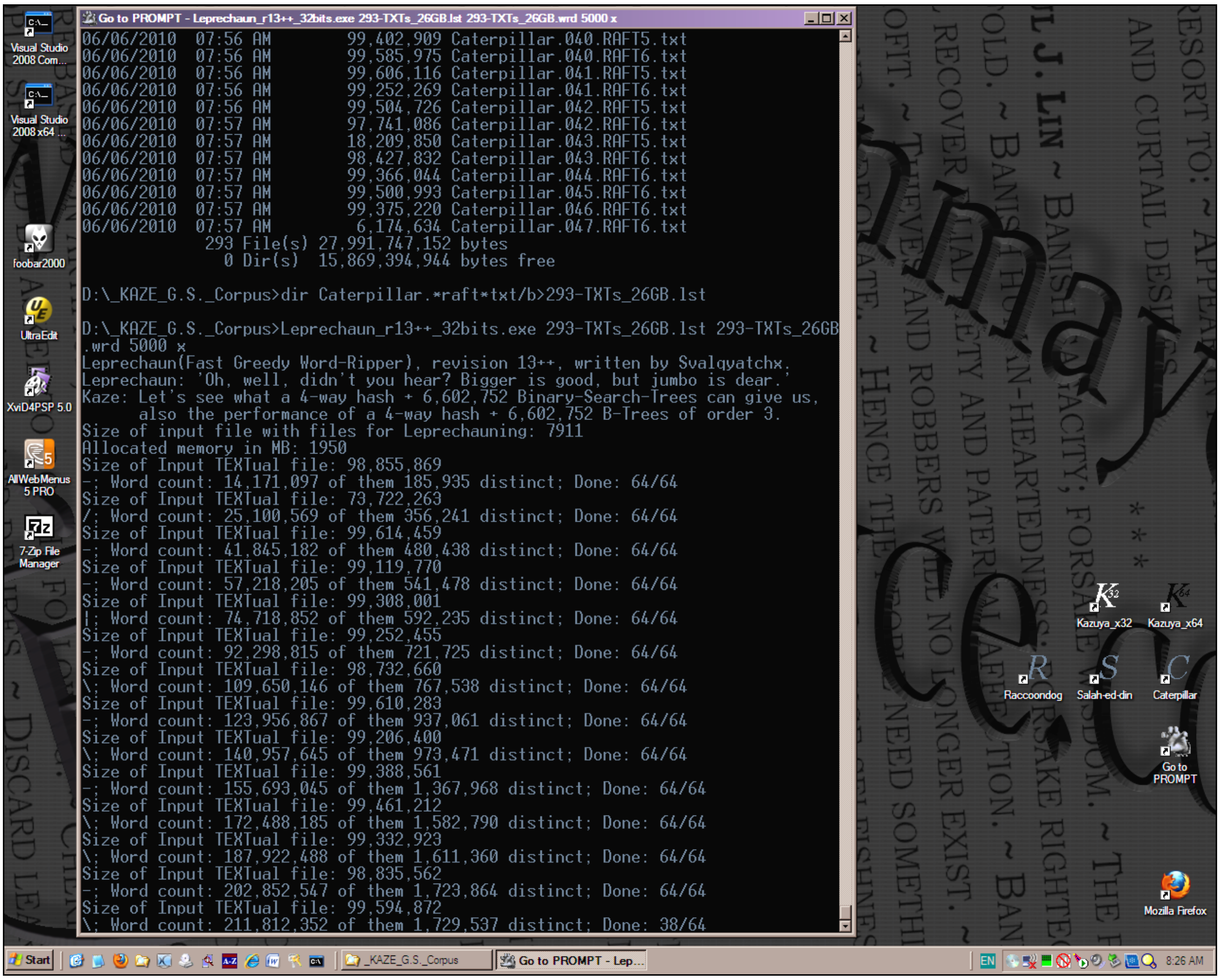
```
Go to PROMPT
06/06/2010  02:55 AM         8,802,963 Caterpillar.043.RAFT5.txt.Okumura
06/05/2010  11:08 PM        36,775,854 Caterpillar.043.RAFT6.txt.gz
06/06/2010  12:38 AM        47,040,471 Caterpillar.043.RAFT6.txt.Lasse
03/14/2009  02:40 AM        21,947,499 Caterpillar.043.RAFT6.txt.lzma
06/06/2010  02:56 AM        49,166,200 Caterpillar.043.RAFT6.txt.Okumura
06/05/2010  11:08 PM        37,484,641 Caterpillar.044.RAFT6.txt.gz
06/06/2010  12:38 AM        47,949,645 Caterpillar.044.RAFT6.txt.Lasse
03/14/2009  02:40 AM        21,386,300 Caterpillar.044.RAFT6.txt.lzma
06/06/2010  02:56 AM        50,045,923 Caterpillar.044.RAFT6.txt.Okumura
06/05/2010  11:08 PM        37,531,406 Caterpillar.045.RAFT6.txt.gz
06/06/2010  12:38 AM        47,993,806 Caterpillar.045.RAFT6.txt.Lasse
03/14/2009  02:40 AM        23,486,608 Caterpillar.045.RAFT6.txt.lzma
06/06/2010  02:56 AM        49,959,768 Caterpillar.045.RAFT6.txt.Okumura
06/05/2010  11:08 PM        37,542,377 Caterpillar.046.RAFT6.txt.gz
06/06/2010  12:38 AM        48,105,477 Caterpillar.046.RAFT6.txt.Lasse
03/14/2009  02:40 AM        22,846,789 Caterpillar.046.RAFT6.txt.lzma
06/06/2010  02:57 AM        50,147,088 Caterpillar.046.RAFT6.txt.Okumura
06/05/2010  11:08 PM         2,350,329 Caterpillar.047.RAFT6.txt.gz
06/06/2010  12:39 AM         3,005,489 Caterpillar.047.RAFT6.txt.Lasse
03/14/2009  02:40 AM         1,374,331 Caterpillar.047.RAFT6.txt.lzma
06/06/2010  02:57 AM         3,131,275 Caterpillar.047.RAFT6.txt.Okumura
            1172 File(s) 40,519,075,490 bytes
               0 Dir(s)   3,443,306,496 bytes free

D:\_KAZE_G.S._Corpus>dir _SearchIntoRafts
 Volume in drive D is H320_Vol5
 Volume Serial Number is 0CB3-C881

 Directory of D:\_KAZE_G.S._Corpus\_SearchIntoRafts

06/04/2010  06:20 AM    <DIR>          .
06/04/2010  06:20 AM    <DIR>          ..
06/04/2010  06:20 AM            86,016 Caterpillar.exe
06/04/2010  06:20 AM         2,150,463 Caterpillar.gif
06/04/2010  06:20 AM               650 Caterpillar_NON.bat
06/04/2010  06:20 AM                18 Caterpillar_NON.ini
06/04/2010  06:20 AM             4,784 Caterpillar_NON.lst
06/04/2010  06:20 AM           165,311 KAZE_G.S._Corpus_'.chm'_Caterpillar.html
06/04/2010  06:20 AM            17,597 KAZE_G.S._Corpus_'.djv-'_Caterpillar.html

06/04/2010  06:20 AM            61,142 KAZE_G.S._Corpus_'.doc'_Caterpillar.html
06/04/2010  06:20 AM        22,754,305 KAZE_G.S._Corpus_'.htm-'_Caterpillar.html

06/04/2010  06:20 AM            32,717 KAZE_G.S._Corpus_'.lit'_Caterpillar.html
06/04/2010  06:20 AM         2,077,267 KAZE_G.S._Corpus_'.pdf'_Caterpillar.html
06/04/2010  06:20 AM            30,941 KAZE_G.S._Corpus_'.rtf'_Caterpillar.html
06/04/2010  06:20 AM        30,124,393 KAZE_G.S._Corpus_'.txt'_Caterpillar.html
06/04/2010  06:20 AM            34,606 Yoshi.exe
06/04/2010  06:00 AM    <DIR>          _RAFTbyRAFT_Description
              14 File(s)     57,540,210 bytes
               3 Dir(s)   3,443,306,496 bytes free

D:\_KAZE_G.S._Corpus>
```

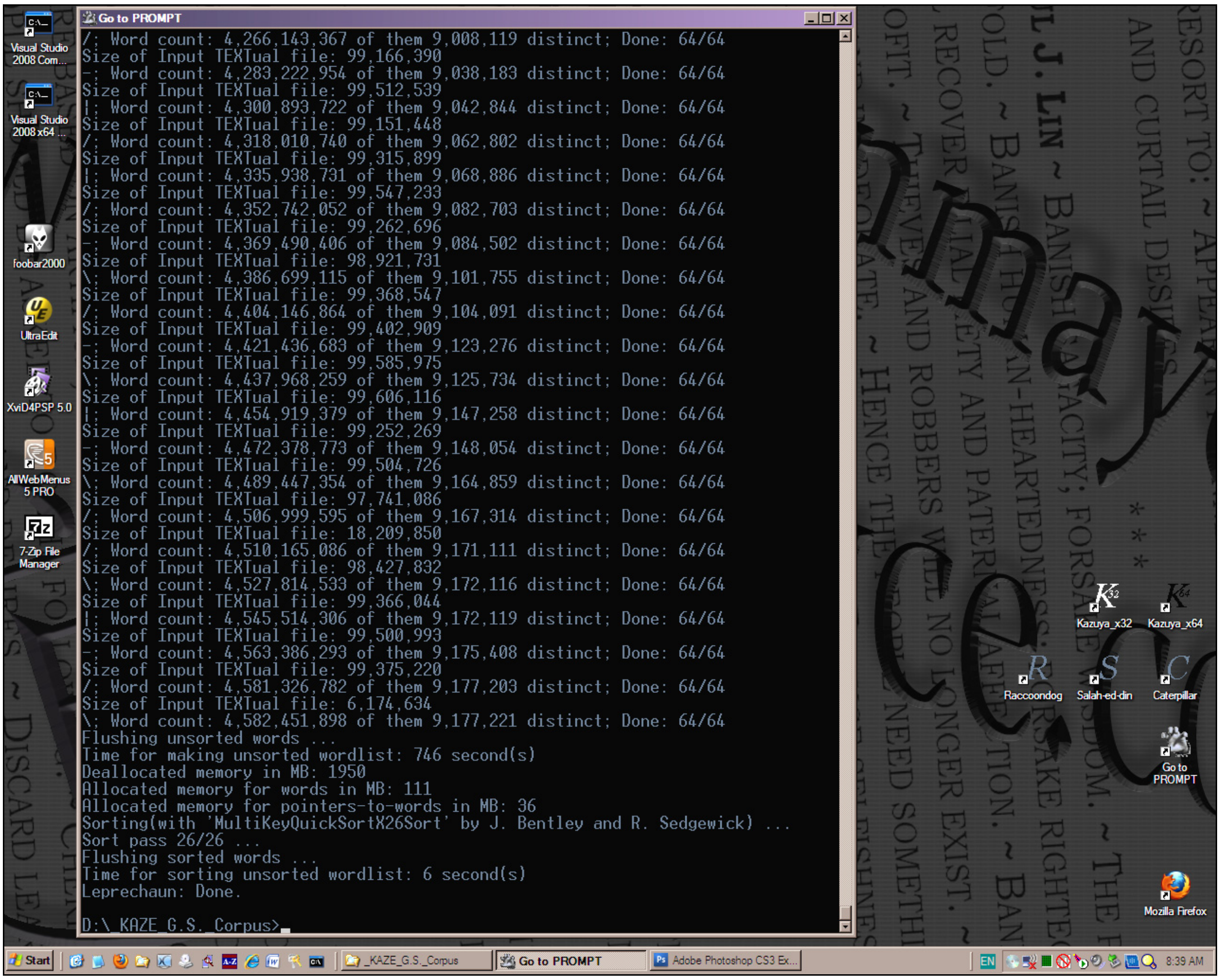

Go to PROMPT
06/04/2010 06:41 AM 19,338 Raccoondog_293_lzma_old.txt
03/14/2009 02:40 AM 682 Raccoondog_HDD2RAM.bat
06/04/2010 06:39 AM 401 Raccoondog_make_md5_CHECK-INTEGRITY.bat
06/06/2010 04:48 AM 164,864 Raccoondog_x64.exe
03/14/2009 02:40 AM 695,326 Salah-ed-din
03/14/2009 02:40 AM 131,072 Salah-ed-din.exe
02/15/2010 05:59 AM 2,167,397 Salah-ed-din.gif
06/06/2010 06:57 AM 92,004 Salah-ed-din.hits.pattern1.html
06/06/2010 06:57 AM 407 Salah-ed-din.hits.pattern2.html
06/06/2010 06:57 AM 407 Salah-ed-din.hits.pattern3.html
06/06/2010 06:57 AM 407 Salah-ed-din.hits.pattern4.html
03/14/2009 02:40 AM 1,078 Salah-ed-din.lnk
06/06/2010 02:59 AM 9,083 Salah-ed-din.lst
06/06/2010 03:15 AM 18,752 Salah-ed-din_293_gz_new.txt
06/06/2010 03:07 AM 18,752 Salah-ed-din_293_gz_old.txt
03/14/2009 02:40 AM 704 Salah-ed-din_HDD2RAM.bat
06/04/2010 06:54 AM 399 Salah-ed-din_make_md5_CHECK-INTEGRITY.bat
03/14/2009 02:40 AM 592,448 Yoshi
03/14/2009 02:40 AM 34,606 Yoshi.exe
06/05/2010 01:45 AM <DIR> _bin
06/06/2010 07:37 AM <DIR> _bin_me
06/04/2010 06:20 AM <DIR> _SearchIntoRafts
663 File(s) 34,334,948,664 bytes
6 Dir(s) 15,752,200,192 bytes free
D:_KAZE_G.S._Corpus>type 293_ALL_lzma_gz_Lasse_Okumura_txt.dir
D:_KAZE_G.S._Corpus>dir Caterpillar.*raft*txt*.lzma
...
03/14/2009 02:40 AM 1,374,331 Caterpillar.047.RAFT6.txt.lzma
293 File(s) 6,214,899,806 bytes
D:_KAZE_G.S._Corpus>dir Caterpillar.*raft*txt*.gz
...
06/05/2010 11:08 PM 2,350,329 Caterpillar.047.RAFT6.txt.gz
293 File(s) 9,470,992,278 bytes
D:_KAZE_G.S._Corpus>dir Caterpillar.*raft*txt*.Lasse
...
06/06/2010 12:39 AM 3,005,489 Caterpillar.047.RAFT6.txt.Lasse
293 File(s) 11,924,741,948 bytes
D:_KAZE_G.S._Corpus>dir Caterpillar.*raft*txt*.Okumura
...
06/06/2010 02:57 AM 3,131,275 Caterpillar.047.RAFT6.txt.Okumura
293 File(s) 12,908,441,458 bytes
D:_KAZE_G.S._Corpus>dir Caterpillar.*raft*txt
...
06/06/2010 07:57 AM 6,174,634 Caterpillar.047.RAFT6.txt
293 File(s) 27,991,747,152 bytes
D:_KAZE_G.S._Corpus>_


Raccoondog
Possible hit: NEW YORK
Pattern(s) example: Pattern1: $$$$$$$$@
Pattern1_NestedPattern1:
Possible hit: Printing
Pattern(s) note: You may specify(four times) a main-pattern(case insensitive
with wildcards '*' i.e. any character(s) or empty, also '@'
i.e. any character or empty, also '#' i.e. any character
and not empty, also '$' i.e. any ALPHA character
and not empty, also '%' i.e. any NON-ALPHA character
and not empty) with three nested-patterns(case
sensitive and unexact), all four connected with AND.
Due to different line endings(CRLF in Windows; LF in UNIX)
you must add a '@' wildcard in place of CR: for example in
case of searching for '*.pdf' write '*.pdf@'.
Pattern(s) example: Pattern1: *%take@%$$@
Pattern1_NestedPattern1:
Possible hit: ... is taken by
Possible hit: ... would take it
Note: First % is to avoid e.g. 'mis' prefix
Second % is to avoid e.g. 'ing' suffix
Master-pattern note: It is case insensitive with wildcards '*','@','#','$','%'
allowed. The purpose of this pattern is to
decide whether a search for next patterns will be
executed, it is applied on all lines i.e. the whole file.
There must be at least one hit in order to execute search
for next patterns.
Have a nice Raccoondoging.
For contacts: sanmayce@hotmail.com
Sanmayce Svalqyatchx 'Kaze', 2010 Jun 06.
Allocated memory for DEcoded file in MB: 256
Size of input file with files for Raccoondoging: 9669
Input Master-pattern(hit only 'Enter' to skip):
Input Pattern1(hit only 'Enter' to skip): *
- Input Pattern1_NestedPattern1(hit only 'Enter' to skip): underdog
- Input Pattern1_NestedPattern2(hit only 'Enter' to skip):
Input Pattern2(hit only 'Enter' to skip):
Processing .\Caterpillar.001.RAFT2.txt.lzma ...
Doing DECODE from HDD to RAM ...
Overall decode performance so far: 000,008KB/clock(EN) or 000,034KB/clock(DE)
Doing SEARCH for Pattern1 at once and flushing hit-sentences ...
000,000,001 The Japanese have long considered Yoshitsune to be the epitome of th
e underdog hero; many legends, stories, kabuki plays, and even films celebrate h
is adventures with his faithful follower, the monk Benkei.
000,000,002 Without Woods, the United States was underdog for the Ryder Cup at V
alhalla in Louisville, Ky.
Found 2 case-sensitive and unexact matches(hits), so far.
Processing .\Caterpillar.002.RAFT2.txt.lzma ...
Doing DECODE from HDD to RAM ...


Raccoondog
erdog will win.
Found 638 case-sensitive and unexact matches(hits), so far.
Processing .\Caterpillar.009.RAFT0.txt.lzma ...
Doing DECODE from HDD to RAM ...
Overall decode performance so far: 000,008KB/clock(EN) or 000,039KB/clock(DE)
Doing SEARCH for Pattern1 at once and flushing hit-sentences ...
Found 638 case-sensitive and unexact matches(hits), so far.
Processing .\Caterpillar.010.RAFT0.txt.lzma ...
Doing DECODE from HDD to RAM ...
Overall decode performance so far: 000,008KB/clock(EN) or 000,039KB/clock(DE)
Doing SEARCH for Pattern1 at once and flushing hit-sentences ...
000,000,639 Just because he is a boss, whatsoever he does looks bad, appears bad, because deep down you resent that you are an underdog and somebody is a topdog.
000,000,640 You resent the fact of being an underdog, so whatsoever is said looks bad, but you cannot react; it will be too costly.
000,000,641 It is creating a division in yourself -- the division of the topdog and the underdog, the division of the superior and the inferior, the division of the saint and the sinner, the division of good and bad the division, basically, of God and devil.
000,000,642 You create a topdog and an underdog, and so there is conflict.
000,000,643 I mean, Bagman continued, lowering his voice still further, you re the underdog here, Harry Anything I can do to help No, said Harry so quickly he knew he had sounded rude, no - I - I know what I m going to do, thanks.
Found 643 case-sensitive and unexact matches(hits), so far.
Processing .\Caterpillar.011.RAFT0.txt.lzma ...
Doing DECODE from HDD to RAM ...
Overall decode performance so far: 000,008KB/clock(EN) or 000,039KB/clock(DE)
Doing SEARCH for Pattern1 at once and flushing hit-sentences ...
000,000,644 They had no conception of the temptations of the underdog.
000,000,645 They had no conception of the temptations of the underdog.
Found 645 case-sensitive and unexact matches(hits), so far.
Total Rough Upload and Decode time: 685,170 clocks
Total Rough Search time: 135,348 clocks
Total time: 823 seconds
Total Lines encountered: 424,754,717
Total Search(non-mask) function invocations: 424,534,498
Total Search(MASK i.e. wildcard) function invocations: 0
Total MASK i.e. wildcard invocations: 0
Total MASK i.e. wildcard hits: 0
Total MASK i.e. wildcard time: 0 clocks
Total BoyerMooreHorspool invocations: 293
Total BoyerMooreHorspool(whole chunks, not lines) hits: 709
Total BoyerMooreHorspool(whole chunks, not lines) time: 18,040 clocks
Total BoyerMooreHorspool(whole chunks, not lines) performance: 1,515KB/clock
Total KarpRabinKaze invocations: 293
Total KarpRabinKaze(whole chunks, not lines) hits: 709
Total KarpRabinKaze(whole chunks, not lines) time: 41,149 clocks
Total KarpRabinKaze(whole chunks, not lines) performance: 664KB/clock
Raccoondog: Done successfully.
Press a key to exit.
Searches into a 'Raccoondog.lst' filelist file! Press 'Esc' to skip rest hits.
Kazuya_x32
Kazuya_x64
Raccoondog
Salah-ed-din
Caterpillar
Go to PROMPT
Mozilla Firefox
Start
_KAZE_G.S._Corpus
Raccoondog
Adobe Photoshop CS3 Ex...
6:42 AM


Raccoondog Hits - Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///D:/_KAZE_G.S._Corpus/Raccoondog.hits.pattern1.html
Yahoo
Book of Lao Zi: Dao De Jing
Nifty console tools and Lao Tzu Tao Te ...
Kazuya Hits
Raccoondog Hits
The Japanese have long considered Yoshitsune to be the epitome of the underdog hero; many legends, stories, kabuki plays, and even films celebrate his advent
Without Woods, the United States was underdog for the Ryder Cup at Valhalla in Louisville, Ky.
It is deeply sympathetic with the psychology of the underdog (1939: 277).
tremendously fascinated by the rise of the various captains of industry and that the underdog always interested me, his needs, his woes, his simplicities.
Although he always felt great sympathy for the underdog, it was only in 1937 in Barcelona, while taking part in the fight against fascism, that he suddenly
His perennial sympathy with the underdog led him into hostility to the upper dog in the modern political world.
Here he gained his distinctive literary character, again and again dealing with people s treachery to each other and with the apparent betrayal of God by me
First, without necessarily subscribing to the patriotic myth of the Greeks as heroic underdogs, it is beyond doubt that, up to the conquests of Alexander, t
Hibbert does not make this zany love for the underdog seadog sound like a good thing.
At the Oratory 263 beginning of his career, of course, he lacked the requisite auctoritas; but in Pro Roscio Amerino (80 BC), a defence on a charge of parri
Churchill never sentimentalizes her underdogs: in near monosyllabic exchanges between three generations of fen women, body-wracking labour and child-rearing
Fermoy wants the Irish to reinvent themselves without pretence to civilization, and without reference to a gentle Christ-figure who accepts pain and champio
, in practice, one might argue that its existence precedes its essence), has found so many niches in popular culture and has become a favored vehicle for as
Hipponax s is an underdog pose, enhanced in this fragment by a physical sketch of the poet shivering, afflicted, and railing against the neglectful god of p
Iambography is an art, not the authentic self-expression of angry underdogs.
As losers in Vietnam, popular culture reinvented Americans as valiant underdogs and, in a paradoxical turn, identified with the little guys against big Evil
As we have seen in this chapter, there is plenty to suggest that Athenian comic poets were quite comfortable as parodists and satirists of tragedy, and that
An activist, a champion of the underdog, and a moralist, Rice advocated William Foster s radical political campaign in 1932, provided financial support for
In defending the underdog, the youngest Magrath simultaneously defends herself.
This apparent lack of interest in her crime and in the welfare of her estranged husband indicates Babe s true interests lay elsewhere in underdogs.
But stop there for a moment; dont give him the advantage of a chance to cloud the issue by that purely automatic sympathy for the underdog simply because he
Seymour Martin Lipset, speaking at the Hebrew University, warned that Israel s biggest problem is likely to be that Shultz is a do-good Christian with sympa
Aftermath Classics in Politics: The Fateful Triangle Noam Chomsky 653 sacre, a feeling among many Israelis that over the years their country had strayed some
"World opinion is always on the side of the underdog.
When they have power, physical, financial or political, neither Hitler nor Stalin have anything on them for cruelty or mistreatment of the underdog.
Goths and other darksiders are often loners and underdogs.
The underdog finds a voice.
Then too, many of them identify with the underdog and when they read the history of the Inquisition and other injustices committed by the Established Author
Picking up strangers as they wander along the path of life, they have the propensity both to care for the underdog, and themselves be victimized.
For John Page, the angel that rode in the whirlwind of the American Revolution and directed the storm indicated that the hand of God was on the side of the
After all, what began in 1939 was only a European war between Poland and, on the other side, Nazi Germany and the Soviet Union, with Britain and France sidi
Like Tolkien's hobbits, the English considered themselves the plucky little underdogs pitted against an all-knowing, all-powerful foe.
* Not * Admittedly, the Americans liked nothing better than to think of themselves as the plucky underdogs.
The engine of history, thus, was moved forward by the dialectical struggle between those who controlled the means, the ruling class, and those in antithetic
Or will it be Johnson and his underdog team of designer Christians?
Gould is a famous defender of underdogs and outcasts, and he deplored the "ritualistic ridicule" (Gould 1982b, p.
3 Citing examples such as how the story of Eve and the serpent is treated, or how the figure of Cain is sometimes elevated, Jonas asserted: This opting for
Hunt argues that these villains are represented in terms of a lurking fascism (the logical conclusion of the authoritarian drift) at odds with Regan s under
5 GOD THE MECHANIC Science loves an underdog, and like any teacher of science, I've got a slew of underdog stories that I relate to my students.
My own gallery of underdog heroes includes Barbara McClintock, whose decades-long studies of corn genetics showed that genes can "jump" from one position to
Each of these discoveries was revolutionary, each directly contradicted prevailing wisdom in the field, and in each case the work of the underdog pushed sci
Does this mean that our current ideas about evolution might be modified, updated, or even thrown out altogether by a crusading underdog?
The latest would-be underdog is Michael Behe, a professor of biochemistry at Lehigh University in Pennsylvania.
Some biologists have toyed with the idea of macromutations, but the topic is still very much the underdog to micromutational studies.
It has been argued that traditions of social justice impel Jewish artists to create imagery of the underdog.
Baer lost the title in his first defense on June 13, 1935, to Jim Braddock, The Cinderella Man, who was listed as a 10 1 underdog.
Despite putting $50 million of his own money into the campaign, Bloomberg was a decided underdog, but he received a last-minute endorsement from Giuliani, wh
I gave praise to Mont, while his people mourned him (Lichtheim in COS) Both Sinuhe and David are underdog warriors who surprisingly vanquish the enemy champ
His writings are marked by a deep compassion for the underdog and a sensitive lyrical quality.
However, while evincing sympathy for the Jews as the historic underdog of Western society, Gandhi was less sympathetic to the Jewish religion.
Find: underdog
Next
Previous
Highlight all
Match case
Done
Start
_KAZE_G.S._Corpus
Raccoondog
Adobe Photoshop CS3 Ex...
Raccoondog Hits - M...
6:47 AM

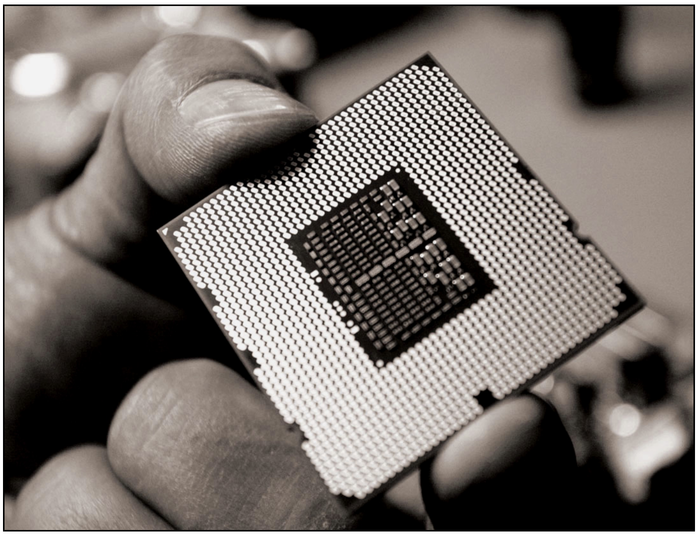

Go to PROMPT - Leprechaun_r13++_32bits.exe 293-TXTs_26GB.lst 293-TXTs_26GB.wrd 5000 x
06/06/2010 07:56 AM   99,402,909 Caterpillar.040.RAFT5.txt
06/06/2010 07:56 AM   99,585,975 Caterpillar.040.RAFT6.txt
06/06/2010 07:56 AM   99,606,116 Caterpillar.041.RAFT5.txt
06/06/2010 07:56 AM   99,252,269 Caterpillar.041.RAFT6.txt
06/06/2010 07:56 AM   99,504,726 Caterpillar.042.RAFT5.txt
06/06/2010 07:57 AM   97,741,086 Caterpillar.042.RAFT6.txt
06/06/2010 07:57 AM   18,209,850 Caterpillar.043.RAFT5.txt
06/06/2010 07:57 AM   98,427,832 Caterpillar.043.RAFT6.txt
06/06/2010 07:57 AM   99,366,044 Caterpillar.044.RAFT6.txt
06/06/2010 07:57 AM   99,500,993 Caterpillar.045.RAFT6.txt
06/06/2010 07:57 AM   99,375,220 Caterpillar.046.RAFT6.txt
06/06/2010 07:57 AM   6,174,634 Caterpillar.047.RAFT6.txt
293 File(s) 27,991,747,152 bytes
0 Dir(s) 15,869,394,944 bytes free
D:_KAZE_G.S._Corpus>dir Caterpillar.*raft*txt/b>293-TXTs_26GB.lst
D:_KAZE_G.S._Corpus>Leprechaun_r13++_32bits.exe 293-TXTs_26GB.lst 293-TXTs_26GB.wrd 5000 x
Leprechaun(Fast Greedy Word-Ripper), revision 13++, written by Svalqyatchx.
Leprechaun: 'Oh, well, didn't you hear? Bigger is good, but jumbo is dear.'
Kaze: Let's see what a 4-way hash + 6,602,752 Binary-Search-Trees can give us,
also the performance of a 4-way hash + 6,602,752 B-Trees of order 3.
Size of input file with files for Leprechauning: 7911
Allocated memory in MB: 1950
Size of Input TEXTual file: 98,855,869
-; Word count: 14,171,097 of them 185,935 distinct; Done: 64/64
Size of Input TEXTual file: 73,722,263
/; Word count: 25,100,569 of them 356,241 distinct; Done: 64/64
Size of Input TEXTual file: 99,614,459
-; Word count: 41,845,182 of them 480,438 distinct; Done: 64/64
Size of Input TEXTual file: 99,119,770
-; Word count: 57,218,205 of them 541,478 distinct; Done: 64/64
Size of Input TEXTual file: 99,308,001
|; Word count: 74,718,852 of them 592,235 distinct; Done: 64/64
Size of Input TEXTual file: 99,252,455
-; Word count: 92,298,815 of them 721,725 distinct; Done: 64/64
Size of Input TEXTual file: 98,732,660
\; Word count: 109,650,146 of them 767,538 distinct; Done: 64/64
Size of Input TEXTual file: 99,610,283
-; Word count: 123,956,867 of them 937,061 distinct; Done: 64/64
Size of Input TEXTual file: 99,206,400
\; Word count: 140,957,645 of them 973,471 distinct; Done: 64/64
Size of Input TEXTual file: 99,388,561
-; Word count: 155,693,045 of them 1,367,968 distinct; Done: 64/64
Size of Input TEXTual file: 99,461,212
\; Word count: 172,488,185 of them 1,582,790 distinct; Done: 64/64
Size of Input TEXTual file: 99,332,923
\; Word count: 187,922,488 of them 1,611,360 distinct; Done: 64/64
Size of Input TEXTual file: 98,835,562
-; Word count: 202,852,547 of them 1,723,864 distinct; Done: 64/64
Size of Input TEXTual file: 99,594,872
\; Word count: 211,812,352 of them 1,729,537 distinct; Done: 38/64
Start
_KAZE_G.S._Corpus
Go to PROMPT - Lep...
8:26 AM


Go to PROMPT
/; Word count: 4,266,143,367 of them 9,008,119 distinct; Done: 64/64
Size of Input TEXTual file: 99,166,390
-; Word count: 4,283,222,954 of them 9,038,183 distinct; Done: 64/64
Size of Input TEXTual file: 99,512,539
|; Word count: 4,300,893,722 of them 9,042,844 distinct; Done: 64/64
Size of Input TEXTual file: 99,151,448
/; Word count: 4,318,010,740 of them 9,062,802 distinct; Done: 64/64
Size of Input TEXTual file: 99,315,899
|; Word count: 4,335,938,731 of them 9,068,886 distinct; Done: 64/64
Size of Input TEXTual file: 99,547,233
/; Word count: 4,352,742,052 of them 9,082,703 distinct; Done: 64/64
Size of Input TEXTual file: 99,262,696
-; Word count: 4,369,490,406 of them 9,084,502 distinct; Done: 64/64
Size of Input TEXTual file: 98,921,731
\; Word count: 4,386,699,115 of them 9,101,755 distinct; Done: 64/64
Size of Input TEXTual file: 99,368,547
/; Word count: 4,404,146,864 of them 9,104,091 distinct; Done: 64/64
Size of Input TEXTual file: 99,402,909
-; Word count: 4,421,436,683 of them 9,123,276 distinct; Done: 64/64
Size of Input TEXTual file: 99,585,975
\; Word count: 4,437,968,259 of them 9,125,734 distinct; Done: 64/64
Size of Input TEXTual file: 99,606,116
|; Word count: 4,454,919,379 of them 9,147,258 distinct; Done: 64/64
Size of Input TEXTual file: 99,252,269
-; Word count: 4,472,378,773 of them 9,148,054 distinct; Done: 64/64
Size of Input TEXTual file: 99,504,726
\; Word count: 4,489,447,354 of them 9,164,859 distinct; Done: 64/64
Size of Input TEXTual file: 97,741,086
/; Word count: 4,506,999,595 of them 9,167,314 distinct; Done: 64/64
Size of Input TEXTual file: 18,209,850
/; Word count: 4,510,165,086 of them 9,171,111 distinct; Done: 64/64
Size of Input TEXTual file: 98,427,832
\; Word count: 4,527,814,533 of them 9,172,116 distinct; Done: 64/64
Size of Input TEXTual file: 99,366,044
|; Word count: 4,545,514,306 of them 9,172,119 distinct; Done: 64/64
Size of Input TEXTual file: 99,500,993
-; Word count: 4,563,386,293 of them 9,175,408 distinct; Done: 64/64
Size of Input TEXTual file: 99,375,220
/; Word count: 4,581,326,782 of them 9,177,203 distinct; Done: 64/64
Size of Input TEXTual file: 6,174,634
\; Word count: 4,582,451,898 of them 9,177,221 distinct; Done: 64/64
Flushing unsorted words ...
Time for making unsorted wordlist: 746 second(s)
Deallocated memory in MB: 1950
Allocated memory for words in MB: 111
Allocated memory for pointers-to-words in MB: 36
Sorting(with 'MultiKeyQuickSortX26Sort' by J. Bentley and R. Sedgewick) ...
Sort pass 26/26 ...
Flushing sorted words ...
Time for sorting unsorted wordlist: 6 second(s)
Leprechaun: Done.
D:_KAZE_G.S._Corpus>
Start
_KAZE_G.S._Corpus
Go to PROMPT
Adobe Photoshop CS3 Ex...
8:39 AM

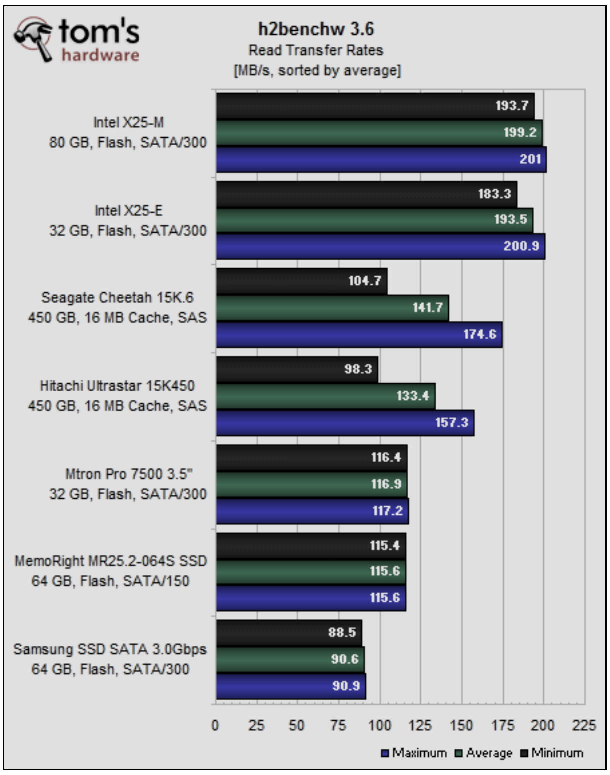
```
D:\_KAZE_G.S._Corpus>type Leprechaun.LOG
Leprechaun report:
A(not always THE) Binary-Search-Tree with the longest path(height, PEAK, number of levels):
  ]sysslade]
     ]swincian]
       [swedloff]
      ]surtaxez[
           ]suddenne]
            ]stongrly]
             [spellchk]
          [spammail[
         [shouzoug[
        [shotaike[
       [shitench[
    ]shahrani[
     [sgcenari]
   ]sessionx[
    [sedanais]
 ]schebaum[ ROOT
  ]scaunele]
     [scappard]
    ]scachans[
     [santinha]
   ]sankhaya[
    ]saleeite]
     [saisihan]
Above Binary-Search-Tree with MaxPEAK = 13 has NODEs = 23 and LEAFs = 7
Legend:
At left side of the word - '[' means no left successor
At left side of the word - ']' means left successor exists
At right side of the word - ']' means no right successor
At right side of the word - '[' means right successor exists
Bytes per second performance: 37,522,449B/s
Words per second performance: 6,142,696W/s
Input File with a list of TEXTual Files: 293-TXTs_26GB.lst
Size of all TEXTual Files: 27,991,747,152
Word count: 4,582,451,898 of them 9,177,221 distinct
Number Of Files: 293
Number Of Lines: 424754717
Allocated memory in MB: 1950
Number Of Trees(GREATER THE BETTER): 2855919
Forest population(Hash Function Quality regarding Collisions i.e. Hash Table Utilization): 43%
Number Of Hash Collisions(Distinct WORDs - Number Of Trees): 6321302
Maximum Attempts to Find/Put a WORD into a Binary-Search-Tree: '13'
Total Attempts to Find/Put WORDs into Binary-Search-Trees: 4,746,283,042
Total Number of LEAFs in Binary-Search-Trees(GREATER THE BETTER): 4,361,992
Perfectly-Balanced-Binary-Search-Tree for MaxNODEs = 34 must have PEAK = 6 = rounding down of integer (1+lb(34))
Binary-Search-Tree(1st out of 1) with MaxNODEs = 34 has PEAK = 11 and LEAFs = 11
Binary-Search-Tree(1st out of 2) with MaxPEAK = '13' has NODEs = 23 and LEAFs = 7
Binary-Search-Tree(1st out of 3) with MaxLEAFs = 12 has NODEs = 27 and PEAK = 8
Words with length 01 occupy 0,033KB of 0,162KB given i.e. 19% utilization
Words with length 02 occupy 0,033KB of 0,162KB given i.e. 19% utilization
Words with length 03 occupy 0,040KB of 0,162KB given i.e. 24% utilization
Words with length 04 occupy 0,158KB of 0,646KB given i.e. 24% utilization
Words with length 05 occupy 0,487KB of 1,775KB given i.e. 27% utilization
Words with length 06 occupy 0,991KB of 3,549KB given i.e. 27% utilization
Words with length 07 occupy 1,431KB of 5,968KB given i.e. 23% utilization
Words with length 08 occupy 1,803KB of 7,581KB given i.e. 23% utilization
Words with length 09 occupy 1,643KB of 8,549KB given i.e. 19% utilization
Words with length 10 occupy 1,546KB of 8,065KB given i.e. 19% utilization
Words with length 11 occupy 1,317KB of 7,420KB given i.e. 17% utilization
Words with length 12 occupy 1,131KB of 6,130KB given i.e. 18% utilization
Words with length 13 occupy 0,945KB of 5,162KB given i.e. 18% utilization
Words with length 14 occupy 0,796KB of 4,033KB given i.e. 19% utilization
Words with length 15 occupy 0,662KB of 3,226KB given i.e. 20% utilization
Words with length 16 occupy 0,561KB of 2,904KB given i.e. 19% utilization
Words with length 17 occupy 0,461KB of 2,259KB given i.e. 20% utilization
Words with length 18 occupy 0,394KB of 1,613KB given i.e. 24% utilization
Words with length 19 occupy 0,335KB of 1,291KB given i.e. 25% utilization
Words with length 20 occupy 0,297KB of 1,130KB given i.e. 26% utilization
Words with length 21 occupy 0,266KB of 0,968KB given i.e. 27% utilization
Words with length 22 occupy 0,248KB of 0,807KB given i.e. 30% utilization
Words with length 23 occupy 0,222KB of 0,646KB given i.e. 34% utilization
Words with length 24 occupy 0,210KB of 0,484KB given i.e. 43% utilization
Words with length 25 occupy 0,194KB of 0,484KB given i.e. 40% utilization
Words with length 26 occupy 0,178KB of 0,323KB given i.e. 55% utilization
Words with length 27 occupy 0,164KB of 0,323KB given i.e. 50% utilization
Words with length 28 occupy 0,160KB of 0,323KB given i.e. 49% utilization
Words with length 29 occupy 0,150KB of 0,323KB given i.e. 46% utilization
Words with length 30 occupy 0,138KB of 0,162KB given i.e. 85% utilization
Words with length 31 occupy 0,134KB of 0,162KB given i.e. 82% utilization
Total pseudo(including hash table) memory utilization: 22%
Total real(wordlist's words VS allocated block) memory utilization: 47/1000
Used value for third parameter in KB: 5000
Use next time as third parameter: 4279-
Time for making unsorted wordlist: 746 second(s)
Time for sorting unsorted wordlist: 6 second(s)
```

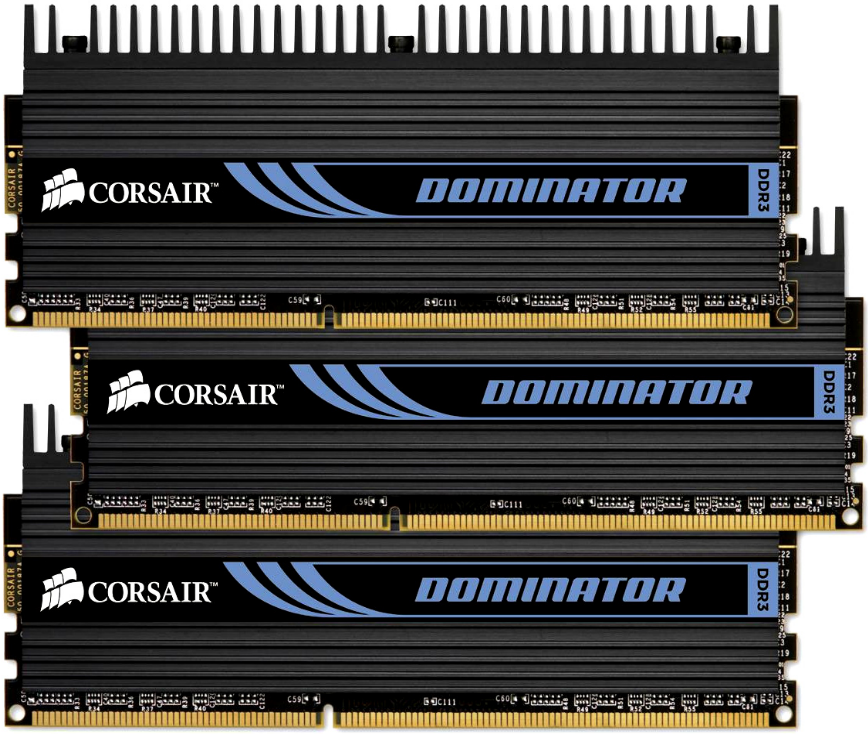
```
#define ulPrime ((unsigned long) 0x00FF00F1)
#define ulBase  ((unsigned long) 127)
         // 9,223,372,036,854,775,807
// 257^7 =     74,051,159,531,521,793
// 257^8 = 19,031,147,999,601,100,801
// 127^9 =  8,594,754,748,609,397,887
// 57^10 =    362,033,331,456,891,249
// 13^16 =    665,416,609,183,179,841
//  5^12 =                244,140,625
// 13^8  =                815,730,721

long KarpRabinKazeHits (char * pbTarget,
     char * pbPattern,
     unsigned long cbTarget,
     unsigned long cbPattern)
{
    unsigned int    i;
    char *  pbTargetMax = pbTarget + cbTarget;
    char *  pbPatternMax = pbPattern + cbPattern;
    unsigned long  ulBaseToPowerMod = 1;
    register unsigned long  ulHashPattern = 0;
    unsigned long  ulHashTarget = 0;
long hits = 0;
//unsigned long count;
    //char *  buf1;
    //char *  buf2;

    if (cbPattern > cbTarget)
        return(0);

    // Compute the power of the left most character in base ulBase
    //for (i = 1; i < cbPattern; i++) ulBaseToPowerMod = (ulBase * ulBaseToPowerMod);

    // Calculate the hash function for the src (and the first dst)
    while (pbPattern < pbPatternMax)
    {
        // Below lines give 366KB/clock for 'underdog':
        //ulHashPattern = (ulHashPattern*ulBase + *pbPattern);
        //ulHashTarget = (ulHashTarget*ulBase + *pbTarget);
        pbPattern++;
        pbTarget++;
    }
        // Below lines give 436KB/clock for 'underdog' + requirement pattern to be 4 chars min.:
        //ulHashPattern = ( (*(long *)(pbPattern-cbPattern)) & 0xffffff00 ) + *(pbPattern-1);
        //ulHashTarget = ( (*(long *)(pbTarget-cbPattern)) & 0xffffff00 ) + *(pbTarget-1);
        // Below lines give 482KB/clock for 'underdog' + requirement pattern to be 2 chars min.:
        //ulHashPattern = ( (*(unsigned short *)(pbPattern-cbPattern)) | *(pbPattern-1) );
        //ulHashTarget = ( (*(unsigned short *)(pbTarget-cbPattern)) | *(pbTarget-1) );
        // Below lines give 482KB/clock for 'underdog' + requirement pattern to be 2 chars min.:
        //ulHashPattern = ( (*(unsigned short *)(pbPattern-cbPattern)) & 0xff00 ) + *(pbPattern-1);
        //ulHashTarget = ( (*(unsigned short *)(pbTarget-cbPattern)) & 0xff00 ) + *(pbTarget-1);
        // Below lines give 605KB/clock for 'underdog' + requirement pattern to be 2 chars min.:
        //ulHashPattern = ( (*(unsigned short *)(pbPattern-cbPattern))<<8 ) + *(pbPattern-1);
        //ulHashTarget = ( (*(unsigned short *)(pbTarget-cbPattern))<<8 ) + *(pbTarget-1);
        // Below lines give 668KB/clock for 'underdog':
        ulHashPattern = ( (*(char *)(pbPattern-cbPattern))<<8 ) + *(pbPattern-1);
        ulHashTarget = ( (*(char *)(pbTarget-cbPattern))<<8 ) + *(pbTarget-1);

    // Dynamically produce hash values for the string as we go
    for ( ;; )
    {
        if ( (ulHashPattern == ulHashTarget) && !memcmpKAZE(pbPattern-cbPattern, pbTarget-cbPattern, (unsigned int)cbPattern) )
       // if ( ulHashPattern == ulHashTarget ) {
       //
       //  count = cbPattern;
       //  buf1 = pbPattern-cbPattern;
       //  buf2 = pbTarget-cbPattern;
       //  while ( --count && *(char *)buf1 == *(char *)buf2 ) {
       //          buf1 = (char *)buf1 + 1;
       //          buf2 = (char *)buf2 + 1;
       //  }
       //
       //  if ( *((unsigned char *)buf1) - *((unsigned char *)buf2) == 0) hits++;
       //  }
            hits++;
            //return((long)(pbTarget-cbPattern));

        if (pbTarget == pbTargetMax)
            return(hits);

        // Below line gives 482KB/clock for 'underdog' + requirement pattern to be 2 chars min.:
        //ulHashTarget = ( (*(unsigned short *)(pbTarget+1-cbPattern)) | *pbTarget );
        // Below line gives 436KB/clock for 'underdog' + requirement pattern to be 4 chars min.:
        //ulHashTarget = ( (*(long *)(pbTarget+1-cbPattern)) & 0xffffff00 ) + *pbTarget;
//; Line 696
//        movsx   esi, BYTE PTR [ebx]
//        mov     ecx, DWORD PTR [edx+1]
//        and     ecx, -256                             ; ffffff00H
//        add     ecx, esi
        // Below line gives 482KB/clock for 'underdog' + requirement pattern to be 2 chars min.:
        //ulHashTarget = ( (*(unsigned short *)(pbTarget+1-cbPattern)) & 0xff00 ) + *pbTarget;
//; Line 691
//        movsx   esi, BYTE PTR [ebx]
//        xor     ecx, ecx
//        mov     cx, WORD PTR [edx+1]
//        and     ecx, 65280                            ; 0000ff00H
//        add     ecx, esi
        // Below line gives 605KB/clock for 'underdog' + requirement pattern to be 2 chars min.:
        //ulHashTarget = ( (*(unsigned short *)(pbTarget+1-cbPattern))<<8 ) + *pbTarget;
        // Below line gives 668KB/clock for 'underdog':
        ulHashTarget = ( (*(char *)(pbTarget+1-cbPattern))<<8 ) + *pbTarget;
//; Line 718
//        movsx   ecx, BYTE PTR [eax+1]
//        movsx   edx, BYTE PTR [ebp]
//        shl     ecx, 8
//        add     ecx, edx
        // Below line gives 366KB/clock for 'underdog':
        //ulHashTarget = (ulHashTarget - *(pbTarget-cbPattern)*ulBaseToPowerMod)*ulBase + *pbTarget;
        pbTarget++;
    }
}
```

Core i7** chip(731 million transistors) – the current dominator; plus **heart-touching-data-storages:

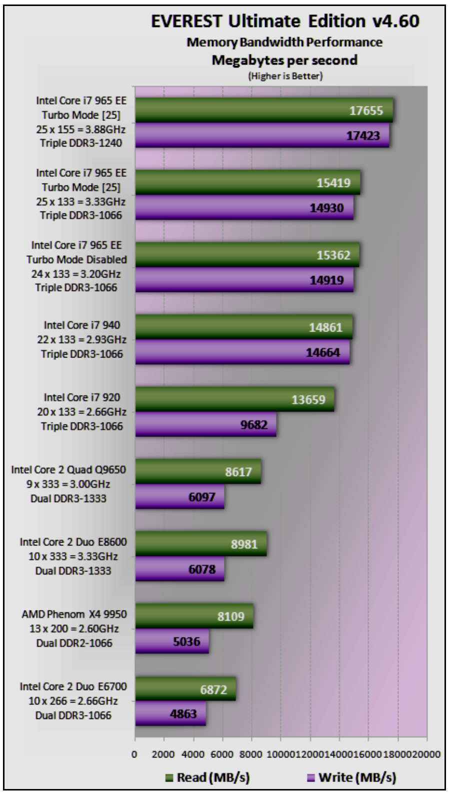


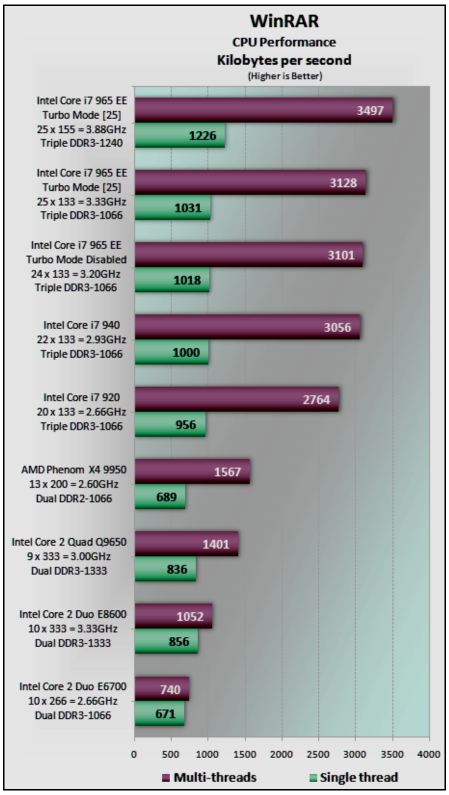
***SSD drives** – simply the future:*

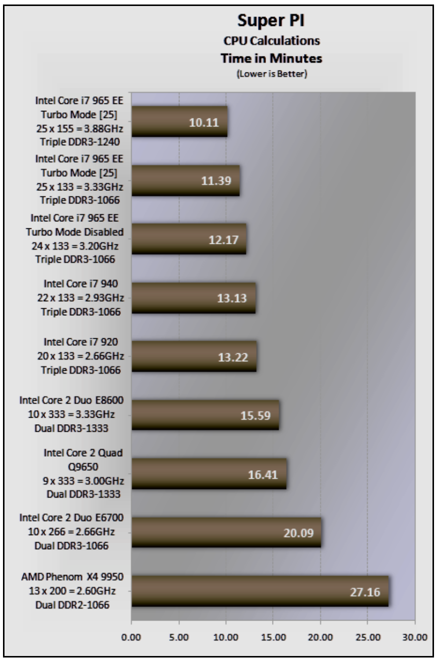


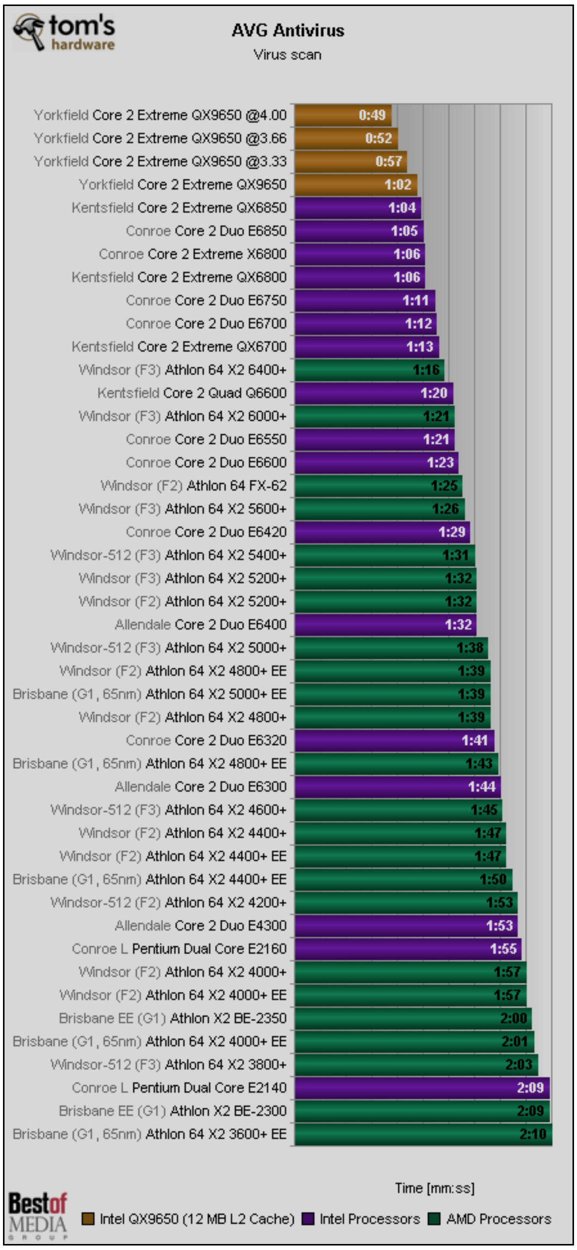

CORSAIR
DOMINATOR
DDR3
CORSAIR
DOMINATOR
DDR3
CORSAIR
DOMINATOR
DDR3

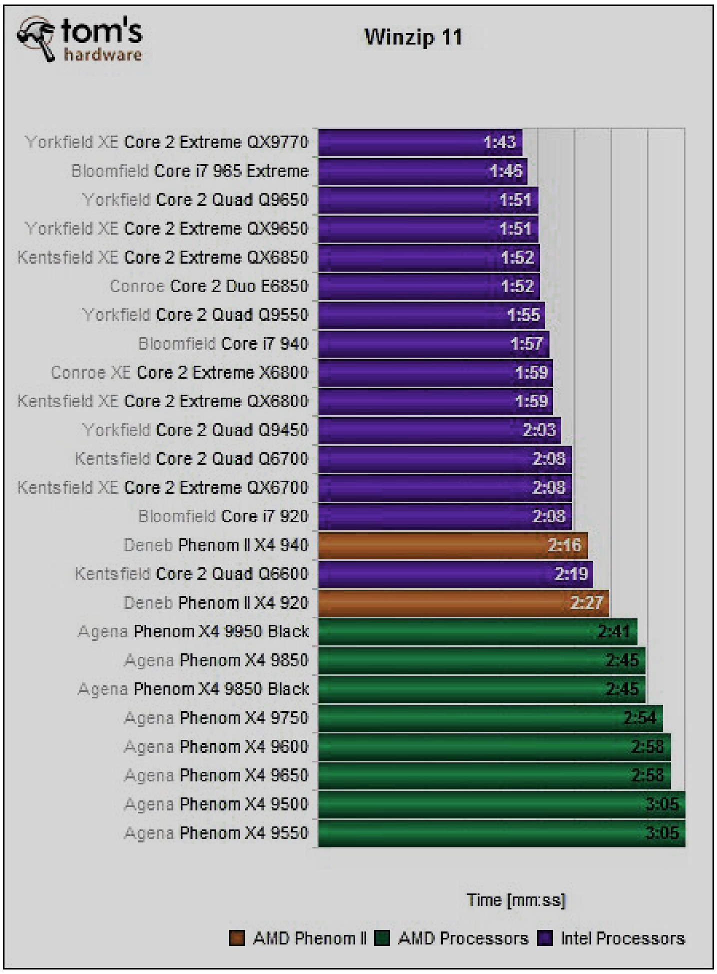

TR3X6G1600C8D
XMS3-1600 2048MB 1600MHz 8-8-8-24 1.65V ver2.1
08420279
CORSAIR


EVEREST Ultimate Edition v4.60
Memory Bandwidth Performance
Megabytes per second
(Higher is Better)
Intel Core i7 965 EE Turbo Mode [25] 25 x 155 = 3.88GHz Triple DDR3-1240
17655
17423
Intel Core i7 965 EE Turbo Mode [25] 25 x 133 = 3.33GHz Triple DDR3-1066
15419
14930
Intel Core i7 965 EE Turbo Mode Disabled 24 x 133 = 3.20GHz Triple DDR3-1066
15362
14919
Intel Core i7 940 22 x 133 = 2.93GHz Triple DDR3-1066
14861
14664
Intel Core i7 920 20 x 133 = 2.66GHz Triple DDR3-1066
13659
9682
Intel Core 2 Quad Q9650 9 x 333 = 3.00GHz Dual DDR3-1333
8617
6097
Intel Core 2 Duo E8600 10 x 333 = 3.33GHz Dual DDR3-1333
8981
6078
AMD Phenom X4 9950 13 x 200 = 2.60GHz Dual DDR2-1066
8109
5036
Intel Core 2 Duo E6700 10 x 266 = 2.66GHz Dual DDR3-1066
6872
4863
0 2000 4000 6000 8000 10000 12000 14000 16000 18000 20000
Read (MB/s)
Write (MB/s)


WinRAR
CPU Performance
Kilobytes per second
(Higher is Better)
Intel Core i7 965 EE Turbo Mode [25] 25 x 155 = 3.88GHz Triple DDR3-1240
3497
1226
Intel Core i7 965 EE Turbo Mode [25] 25 x 133 = 3.33GHz Triple DDR3-1066
3128
1031
Intel Core i7 965 EE Turbo Mode Disabled 24 x 133 = 3.20GHz Triple DDR3-1066
3101
1018
Intel Core i7 940 22 x 133 = 2.93GHz Triple DDR3-1066
3056
1000
Intel Core i7 920 20 x 133 = 2.66GHz Triple DDR3-1066
2764
956
AMD Phenom X4 9950 13 x 200 = 2.60GHz Dual DDR2-1066
1567
689
Intel Core 2 Quad Q9650 9 x 333 = 3.00GHz Dual DDR3-1333
1401
836
Intel Core 2 Duo E8600 10 x 333 = 3.33GHz Dual DDR3-1333
1052
856
Intel Core 2 Duo E6700 10 x 266 = 2.66GHz Dual DDR3-1066
740
671
0 500 1000 1500 2000 2500 3000 3500 4000
Multi-threads
Single thread


Super PI
CPU Calculations
Time in Minutes
(Lower is Better)
Intel Core i7 965 EE Turbo Mode [25] 25 x 155 = 3.88GHz Triple DDR3-1240
10.11
Intel Core i7 965 EE Turbo Mode [25] 25 x 133 = 3.33GHz Triple DDR3-1066
11.39
Intel Core i7 965 EE Turbo Mode Disabled 24 x 133 = 3.20GHz Triple DDR3-1066
12.17
Intel Core i7 940 22 x 133 = 2.93GHz Triple DDR3-1066
13.13
Intel Core i7 920 20 x 133 = 2.66GHz Triple DDR3-1066
13.22
Intel Core 2 Duo E8600 10 x 333 = 3.33GHz Dual DDR3-1333
15.59
Intel Core 2 Quad Q9650 9 x 333 = 3.00GHz Dual DDR3-1333
16.41
Intel Core 2 Duo E6700 10 x 266 = 2.66GHz Dual DDR3-1066
20.09
AMD Phenom X4 9950 13 x 200 = 2.60GHz Dual DDR2-1066
27.16
0.00 5.00 10.00 15.00 20.00 25.00 30.00


tom's hardware
AVG Antivirus
Virus scan
Yorkfield Core 2 Extreme QX9650 @4.00 0:49
Yorkfield Core 2 Extreme QX9650 @3.66 0:52
Yorkfield Core 2 Extreme QX9650 @3.33 0:57
Yorkfield Core 2 Extreme QX9650 1:02
Kentsfield Core 2 Extreme QX6850 1:04
Conroe Core 2 Duo E6850 1:05
Conroe Core 2 Extreme X6800 1:06
Kentsfield Core 2 Extreme QX6800 1:06
Conroe Core 2 Duo E6750 1:11
Conroe Core 2 Duo E6700 1:12
Kentsfield Core 2 Extreme QX6700 1:13
Windsor (F3) Athlon 64 X2 6400+ 1:16
Kentsfield Core 2 Quad Q6600 1:20
Windsor (F3) Athlon 64 X2 6000+ 1:21
Conroe Core 2 Duo E6550 1:21
Conroe Core 2 Duo E6600 1:23
Windsor (F2) Athlon 64 FX-62 1:25
Windsor (F3) Athlon 64 X2 5600+ 1:26
Conroe Core 2 Duo E6420 1:29
Windsor-512 (F3) Athlon 64 X2 5400+ 1:31
Windsor (F3) Athlon 64 X2 5200+ 1:32
Windsor (F2) Athlon 64 X2 5200+ 1:32
Allendale Core 2 Duo E6400 1:32
Windsor-512 (F3) Athlon 64 X2 5000+ 1:38
Windsor (F2) Athlon 64 X2 4800+ EE 1:39
Brisbane (G1, 65nm) Athlon 64 X2 5000+ EE 1:39
Windsor (F2) Athlon 64 X2 4800+ 1:39
Conroe Core 2 Duo E6320 1:41
Brisbane (G1, 65nm) Athlon 64 X2 4800+ EE 1:43
Allendale Core 2 Duo E6300 1:44
Windsor-512 (F3) Athlon 64 X2 4600+ 1:45
Windsor (F2) Athlon 64 X2 4400+ 1:47
Windsor (F2) Athlon 64 X2 4400+ EE 1:47
Brisbane (G1, 65nm) Athlon 64 X2 4400+ EE 1:50
Windsor-512 (F2) Athlon 64 X2 4200+ 1:53
Allendale Core 2 Duo E4300 1:53
Conroe L Pentium Dual Core E2160 1:55
Windsor (F2) Athlon 64 X2 4000+ 1:57
Windsor (F2) Athlon 64 X2 4000+ EE 1:57
Brisbane EE (G1) Athlon X2 BE-2350 2:00
Brisbane (G1, 65nm) Athlon 64 X2 4000+ EE 2:01
Windsor-512 (F3) Athlon 64 X2 3800+ 2:03
Conroe L Pentium Dual Core E2140 2:09
Brisbane EE (G1) Athlon X2 BE-2300 2:09
Brisbane (G1, 65nm) Athlon 64 X2 3600+ EE 2:10
Time [mm:ss]
Best of MEDIA
Intel QX9650 (12 MB L2 Cache)
Intel Processors
AMD Processors


tom's hardware
Winzip 11
Yorkfield XE Core 2 Extreme QX9770 1:43
Bloomfield Core i7 965 Extreme 1:45
Yorkfield Core 2 Quad Q9650 1:51
Yorkfield XE Core 2 Extreme QX9650 1:51
Kentsfield XE Core 2 Extreme QX6850 1:52
Conroe Core 2 Duo E6850 1:52
Yorkfield Core 2 Quad Q9550 1:55
Bloomfield Core i7 940 1:57
Conroe XE Core 2 Extreme X6800 1:59
Kentsfield XE Core 2 Extreme QX6800 1:59
Yorkfield Core 2 Quad Q9450 2:03
Kentsfield Core 2 Quad Q6700 2:08
Kentsfield XE Core 2 Extreme QX6700 2:08
Bloomfield Core i7 920 2:08
Deneb Phenom II X4 940 2:16
Kentsfield Core 2 Quad Q6600 2:19
Deneb Phenom II X4 920 2:27
Agena Phenom X4 9950 Black 2:41
Agena Phenom X4 9850 2:45
Agena Phenom X4 9850 Black 2:45
Agena Phenom X4 9750 2:54
Agena Phenom X4 9600 2:58
Agena Phenom X4 9650 2:58
Agena Phenom X4 9500 3:05
Agena Phenom X4 9550 3:05
Time [mm:ss]
AMD Phenom II
AMD Processors
Intel Processors

CPUs – gone with the wind like an old newspaper:

tom's hardware

Dual Socket F (1207 FX) memory interface : 2x dual channel DDR2-800

CPU-Name	Clock	Multi	FSB / Hyper Tras.	Cache L1/L2	Voltage	Instruction sets	Energy Features	Power	Temp.	Transistors	Core	Step.	Process
Athlon 64 FX-74 (4x4)	4x 3000 MHz	15x free	200 MHz / HT1000	4x 64+64KB / 1MB	1.35 - 1.40V	MMX 3DNow! NX SSE SSE2 SSE3 x86-64 PA	Cool'n'Quiet	250 W	56°C	454 Mio.	Windsor FX	F3	90 nm
Athlon 64 FX-72 (4x4)	4x 2800 MHz	14x free	200 MHz / HT1000	4x 64+64KB / 1MB	1.35 - 1.40V	MMX 3DNow! NX SSE SSE2 SSE3 x86-64 PA	Cool'n'Quiet	250 W	63°C	454 Mio.	Windsor FX	F3	90 nm
Athlon 64 FX-70 (4x4)	4x 2600 MHz	13x free	200 MHz / HT1000	4x 64+64KB / 1MB	1.35 - 1.40V	MMX 3DNow! NX SSE SSE2 SSE3 x86-64 PA	Cool'n'Quiet	250 W	63°C	454 Mio.	Windsor FX	F3	90 nm

Socket AM2 / AM2+ (940 Pins) memory interface : dual channel DDR2-800 / 1066

CPU-Name	Clock	Multi	FSB / Hyper Tras.	Cache L1/L2	Voltage	Instruction sets	Energy Features	Power	Temp.	Transistors	Core	Step.	Process
Phenom X4 9950 Black	4x 2600 MHz	13x open	200 MHz / HT2000	4x 64+64KB / 4x 512 KB / 2 MB	1.05 - 1.30V	MMX 3DNow! NX SSE SSE2 SSE3 SSE4A x86-64 PA	Cool'n'Quiet 2	140 W	64°C	463 Mio.	Agena	B3	65 nm
Phenom X4 9850 Black	4x 2500 MHz	12.5x open	200 MHz / HT2000	4x 64+64KB / 4x 512 KB / 2 MB	1.05 - 1.3V	MMX 3DNow! NX SSE SSE2 SSE3 SSE4A x86-64 PA	Cool'n'Quiet 2	125 W	61°C	463 Mio.	Agena	B3	65 nm
Phenom X4 9850	4x 2500 MHz	12.5x	200 MHz / HT2000	4x 64+64KB / 4x 512 KB / 2 MB	1.05 - 1.3V	MMX 3DNow! NX SSE SSE2 SSE3 SSE4A x86-64 PA	Cool'n'Quiet 2	125 W	61°C	463 Mio.	Agena	B3	65 nm
Phenom X4 9750	4x 2400 MHz	12x	200 MHz / HT2000	4x 64+64KB / 4x 512 KB / 2 MB	1.20/1.25/1.30V	MMX 3DNow! NX SSE SSE2 SSE3 SSE4A x86-64 PA	Cool'n'Quiet 2	125 W	61°C	463 Mio.	Agena	B3	65 nm
Phenom X4 9750	4x 2400 MHz	12x	200 MHz / HT2000	4x 64+64KB / 4x 512 KB / 2 MB	1.10/1.15/1.20/1.25V	MMX 3DNow! NX SSE SSE2 SSE3 SSE4A x86-64 PA	Cool'n'Quiet 2	95 W	70 °C	463 Mio.	Agena	B3	65 nm
Phenom X4 9650	4x 2300 MHz	11.5x	200 MHz / HT1800	4x 64+64KB / 4x 512 KB / 2 MB	1.10/1.15/1.20/1.25V	MMX 3DNow! NX SSE SSE2 SSE3 SSE4A x86-64 PA	Cool'n'Quiet 2	95 W	70 °C	463 Mio.	Agena	B3	65 nm
Phenom X4 9550	4x 2200 MHz	11x	200 MHz / HT1800	4x 64+64KB / 4x 512 KB / 2 MB	1.10/1.15/1.20/1.25V	MMX 3DNow! NX SSE SSE2 SSE3 SSE4A x86-64 PA	Cool'n'Quiet 2	95 W	70 °C	463 Mio.	Agena	B3	65 nm
Phenom X4 9350e	4x 2000 MHz	10x	200 MHz / HT1800	4x 64+64KB / 4x 512 KB / 2 MB	1.0 - 1.125V	MMX 3DNow! NX SSE SSE2 SSE3 SSE4A x86-64 PA	Cool'n'Quiet 2	65 W	70 °C	463 Mio.	Agena	B3	65 nm
Phenom X4 9150e	4x 1800 MHz	9x	200 MHz / HT1800	4x 64+64KB / 4x 512 KB / 2 MB	1.0 - 1.125V	MMX 3DNow! NX SSE SSE2 SSE3 SSE4A x86-64 PA	Cool'n'Quiet 2	65 W	70 °C	463 Mio.	Agena	B3	65 nm
Phenom X4 9100e	4x 1800 MHz	9x	200 MHz / HT1800	4x 64+64KB / 4x 512 KB / 2 MB	1.10/1.125/1.15V	MMX 3DNow! NX SSE SSE2 SSE3 SSE4A x86-64 PA	Cool'n'Quiet 2	65 W	61°C	463 Mio.	Agena	B2	65 nm
Phenom X4 9700 Sample	4x 2400 MHz	12x	200 MHz / HT2000	4x 64+64KB / 4x 512 KB / 2 MB	unknown	MMX 3DNow! NX SSE SSE2 SSE3 SSE4A x86-64 PA	Cool'n'Quiet 2	95 W	unknown	463 Mio.	Agena	B2	65 nm
Phenom X4 9600 Black	4x 2300 MHz	11.5x open	200 MHz / HT1800	4x 64+64KB / 4x 512 KB / 2 MB	1.10/1.15/1.20/1.25V	MMX 3DNow! NX SSE SSE2 SSE3 SSE4A x86-64 PA	Cool'n'Quiet 2	95 W	70 °C	463 Mio.	Agena	B2	65 nm
Phenom X4 9600	4x 2300 MHz	11.5x	200 MHz / HT1800	4x 64+64KB / 4x 512 KB / 2 MB	1.10/1.15/1.20/1.25V	MMX 3DNow! NX SSE SSE2 SSE3 SSE4A x86-64 PA	Cool'n'Quiet 2	95 W	70 °C	463 Mio.	Agena	B2	65 nm
Phenom X4 9500	4x 2200 MHz	11x	200 MHz / HT1800	4x 64+64KB / 4x 512 KB / 2 MB	1.10/1.15/1.20/1.25V	MMX 3DNow! NX SSE SSE2 SSE3 SSE4A x86-64 PA	Cool'n'Quiet 2	95 W	70 °C	463 Mio.	Agena	B2	65 nm
Phenom X3 8750	3x 2400 MHz	12x	200 MHz / HT1800	3x 64+64KB / 3x 512 KB / 2 MB	1.05V - 1.25V	MMX 3DNow! NX SSE SSE2 SSE3 SSE4A x86-64 PA	Cool'n'Quiet 2	95 W	70 °C		Toliman	B3	65 nm
Phenom X3 8650	3x 2300 MHz	11.5x	200 MHz / HT1800	3x 64+64KB / 3x 512 KB / 2 MB	1.05V - 1.25V	MMX 3DNow! NX SSE SSE2 SSE3 SSE4A x86-64 PA	Cool'n'Quiet 2	95 W	70 °C		Toliman	B3	65 nm
Phenom X3 8600	3x 2300 MHz	11.5x	200 MHz / HT1800	3x 64+64KB / 3x 512 KB / 2 MB	1.05V - 1.25V	MMX 3DNow! NX SSE SSE2 SSE3 SSE4A x86-64 PA	Cool'n'Quiet 2	95 W	70 °C		Toliman	B2	65 nm
Phenom X3 8450	3x 2100 MHz	10.5x	200 MHz / HT1800	3x 64+64KB / 3x 512 KB / 2 MB	1.05V - 1.25V	MMX 3DNow! NX SSE SSE2 SSE3 SSE4A x86-64 PA	Cool'n'Quiet 2	95 W	70 °C		Toliman	B3	65 nm
Phenom X3 8400	3x 2100 MHz	10.5x	200 MHz / HT1800	3x 64+64KB / 3x 512 KB / 2 MB	1.05V - 1.25V	MMX 3DNow! NX SSE SSE2 SSE3 SSE4A x86-64 PA	Cool'n'Quiet 2	95 W	70 °C		Toliman	B2	65 nm
Athlon 64 X2 6500+	2x 2300 MHz	11,5x		2x 64+64KB / 2x 512 KB / 2 MB		MMX 3DNow! NX SSE SSE2 SSE3 SSE4A x86-64 PA	Cool'n'Quiet 2	95 W			Kuma	B3	65 nm
Athlon 64 FX FX-62	2x 2800 MHz	14x free	200 MHz / HT1000	2x 64+64KB / 1MB	1.35 - 1.40V	MMX 3DNow! NX SSE SSE2 SSE3 x86-64 PA	Cool'n'Quiet	125 W	55-63°C	227 Mio.	Windsor	F2	90 nm
Athlon 64 X2 6400+ Black	2x 3200 MHz	16x	200 MHz / HT1000	2x 64+64KB / 1MB	1.35-1.40V	MMX 3DNow! NX SSE SSE2 SSE3 x86-64 PA	Cool'n'Quiet	125 W	55-63°C	227 Mio.	Windsor	F3	90 nm
Athlon 64 X2 6000+	2x 3000 MHz	15x	200 MHz / HT1000	2x 64+64KB / 1MB	1.35 - 1.40V	MMX 3DNow! NX SSE SSE2 SSE3 x86-64 PA	Cool'n'Quiet	125 W	55-63°C	227 Mio.	Windsor	F3	90 nm
Athlon 64 X2 5600+	2x 2800 MHz	14x	200 MHz / HT1000	2x 64+64KB / 1MB	1.30 - 1.35V	MMX 3DNow! NX SSE SSE2 SSE3 x86-64 PA	Cool'n'Quiet	89 W	55-70°C	227 Mio.	Windsor	F3	90 nm
Athlon 64 X2 5400+ Black	2x 2800 MHz	14x	200 MHz / HT1000	2x 64+64KB / 512 KB	1.30 - 1.35V	MMX 3DNow! NX SSE SSE2 SSE3 x86-64 PA	Cool'n'Quiet	89 W	55-70°C	154 Mio.	Windsor-512	F3	90 nm
Athlon 64 X2 5400+	2x 2800 MHz	14x	200 MHz / HT1000	2x 64+64KB / 512 KB	1.30 - 1.35V	MMX 3DNow! NX SSE SSE2 SSE3 x86-64 PA	Cool'n'Quiet	89 W	55-70°C	154 Mio.	Windsor-512	F3	90 nm
Athlon 64 X2 5200+	2x 2600 MHz	13x	200 MHz / HT1000	2x 64+64KB / 1MB	1.30V / 1.35V	MMX 3DNow! NX SSE SSE2 SSE3 x86-64 PA	Cool'n'Quiet	89 W	55-70°C	227 Mio.	Windsor	F2	90 nm
Athlon 64 X2 5000+ Black	2x 2600 MHz	13x	200 MHz / HT1000	2x 64+64KB / 512 KB	1.30V / 1.35V	MMX 3DNow! NX SSE SSE2 SSE3 x86-64 PA	Cool'n'Quiet	89 W	55-70°C	154 Mio.	Windsor-512	F2	90 nm
Athlon 64 X2 5000+	2x 2600 MHz	13x	200 MHz / HT1000	2x 64+64KB / 512 KB	1.30V / 1.35V	MMX 3DNow! NX SSE SSE2 SSE3 x86-64 PA	Cool'n'Quiet	89 W	55-70°C	154 Mio.	Windsor-512	F2	90 nm
Athlon 64 X2 5000+	2x 2600 MHz	13x	200 MHz / HT1000	2x 64+64KB / 512 KB	1.30V / 1.35V	MMX 3DNow! NX SSE SSE2 SSE3 x86-64 PA	Cool'n'Quiet	89 W	55-70°C	154 Mio.	Windsor-512	F2	90 nm
Athlon 64 X2 4800+	2x 2400 MHz	12x	200 MHz / HT1000	2x 64+64KB / 1MB	1.30V / 1.35V	MMX 3DNow! NX SSE SSE2 SSE3 x86-64 PA	Cool'n'Quiet	89 W	55-70°C	227 Mio.	Windsor	F2	90 nm
Athlon 64 X2 4600+	2x 2400 MHz	12x	200 MHz / HT1000	2x 64+64KB / 512 KB	1.30V / 1.35V	MMX 3DNow! NX SSE SSE2 SSE3 x86-64 PA	Cool'n'Quiet	89 W	55-70°C	154 Mio.	Windsor-512	F2	90 nm
Athlon 64 X2 4400+	2x 2200 MHz	11x	200 MHz / HT1000	2x 64+64KB / 1MB	1.30V / 1.35V	MMX 3DNow! NX SSE SSE2 SSE3 x86-64 PA	Cool'n'Quiet	89 W	55-70°C	227 Mio.	Windsor	F2	90 nm
Athlon 64 X2 4200+	2x 2200 MHz	11x	200 MHz / HT1000	2x 64+64KB / 512 KB	1.30V / 1.35V	MMX 3DNow! NX SSE SSE2 SSE3 x86-64 PA	Cool'n'Quiet	89 W	55-72°C	154 Mio.	Windsor-512	F2	90 nm
Athlon 64 X2 4000+	2x 2000 MHz	10x	200 MHz / HT1000	2x 64+64KB / 1MB	1.30V / 1.35V	MMX 3DNow! NX SSE SSE2 SSE3 x86-64 PA	Cool'n'Quiet	89 W	55-70°C	227 Mio.	Windsor	F2	90 nm
Athlon 64 X2 3800+	2x 2000 MHz	10x	200 MHz / HT1000	2x 64+64KB / 512 KB	1.30V / 1.35V	MMX 3DNow! NX SSE SSE2 SSE3 x86-64 PA	Cool'n'Quiet	89 W	55-70°C	154 Mio.	Windsor-512	F2	90 nm

tom's hardware

Socket 771 memory interface : dual / quad channel FB-DIMM DDR3-667/800

CPU-Name	Clock	Multi	FSB	Cache L1/L2	Voltage	Instruction sets	Energy Features	Power	Temp.	Trans.	Core	Stepping	Process
Core 2 Extreme QX9775	4x 3200 MHz	8x (open)	400 MHz QDR	4x 32+32 / 2x 6144 KB	1.212 V	MMX SSE SSE2 SSE3 SSSE3 SSE4.1 NX EM64T VT	TM2 C1E EIST	150 W	63°C	820 Mio.	Yorkfield XE	C0	45 nm

Socket 775 memory interface : dual channel DDR2-533/667/800 DDR3-1066/1333

CPU-Name	Clock	Multi	FSB	Cache L1/L2	Voltage	Instruction sets	Energy Features	Power	Temp.	Trans.	Core	Stepping	Process
Core 2 Extreme QX9770	4x 3200 MHz	free (8x)	400 MHz QDR	4x 32+32 / 2x 6144 KB	1.3625 V	MMX SSE SSE2 SSE3 SSSE3 SSE4.1 NX EM64T VT	TM2 C1E EIST	136 W	55.5°C	820 Mio.	Yorkfield XE	C1	45 nm
Core 2 Extreme QX9650	4x 3000 MHz	9x (open)	333 MHz QDR	4x 32+32 / 2x 6144 KB	1.3625 V	MMX SSE SSE2 SSE3 SSSE3 SSE4.1 NX EM64T VT	TM2 C1E EIST	130 W	64.5°C	820 Mio.	Yorkfield XE	C0 C1	45 nm
Core 2 Quad Q9650	4x 3000 MHz	9x	333 MHz QDR	4x 32+32 / 2x 6144 KB		MMX SSE SSE2 SSE3 SSSE3 SSE4.1 NX EM64T VT	TM2 C1E EIST	95 W		820 Mio.	Yorkfield	C1	45 nm
Core 2 Quad Q9550	4x 2833 MHz	8.5x	333 MHz QDR	4x 32+32 / 2x 6144 KB	1.3625 V	MMX SSE SSE2 SSE3 SSSE3 SSE4.1 NX EM64T VT	TM2 C1E EIST	95 W	71.4°C	820 Mio.	Yorkfield	C1	45 nm
Core 2 Quad Q9450	4x 2666 MHz	8x	333 MHz QDR	4x 32+32 / 2x 6144 KB	1.3625 V	MMX SSE SSE2 SSE3 SSSE3 SSE4.1 NX EM64T VT	TM2 C1E EIST	95 W	71.4°C	820 Mio.	Yorkfield	C1	45 nm
Core 2 Quad Q9400	4x 2666 MHz	8x	333 MHz QDR	4x 32+32 / 2x 3072 KB		MMX SSE SSE2 SSE3 SSSE3 SSE4.1 NX EM64T VT	TM2 C1E EIST	95 W			Yorkfield	C1	45 nm
Core 2 Quad Q9300	4x 2500 MHz	7.5x	333 MHz QDR	4x 32+32 / 2x 3072 KB	1.3625 V	MMX SSE SSE2 SSE3 SSSE3 SSE4.1 NX EM64T VT	TM2 C1E EIST	95 W	71.4°C		Yorkfield	M1	45 nm
Core 2 Quad Q8200	4x 2333 MHz	7x	333 MHz QDR	4x 32+32 / 2x 2048 KB		MMX SSE SSE2 SSE3 SSSE3 SSE4.1 NX EM64T VT	TM2 C1E EIST	95 W			Yorkfield	M1	45 nm
Core 2 Duo E8600	2x 3333 MHz	10x	333 MHz QDR	2x 32+32 / 1x 6144 KB		MMX SSE SSE2 SSE3 SSSE3 SSE4.1 NX EM64T VT	TM2 C1E EIST	65 W		410 Mio.	Wolfdale	E0	45 nm
Core 2 Duo E8500	2x 3166 MHz	9.5x	333 MHz QDR	2x 32+32 / 1x 6144 KB	1.3625 V	MMX SSE SSE2 SSE3 SSSE3 SSE4.1 NX EM64T VT	TM2 C1E EIST	65 W	72.4°C	410 Mio.	Wolfdale	C0	45 nm
Core 2 Duo E8400	2x 3000 MHz	9x	333 MHz QDR	2x 32+32 / 1x 6144 KB	1.3625 V	MMX SSE SSE2 SSE3 SSSE3 SSE4.1 NX EM64T VT	TM2 C1E EIST	65 W	72.4°C	410 Mio.	Wolfdale	C0	45 nm
Core 2 Duo E8300	2x 2833 MHz	8.5x	333 MHz QDR	2x 32+32 / 1x 6144 KB	1.3625 V	MMX SSE SSE2 SSE3 SSSE3 SSE4.1 NX EM64T VT	TM2 C1E EIST	65 W	72.4°C	410 Mio.	Wolfdale	C0	45 nm
Core 2 Duo E8200	2x 2666 MHz	8x	333 MHz QDR	2x 32+32 / 1x 6144 KB	1.3625 V	MMX SSE SSE2 SSE3 SSSE3 SSE4.1 NX EM64T VT	TM2 C1E EIST	65 W	72.4°C	410 Mio.	Wolfdale	C0	45 nm
Core 2 Duo E8190	2x 2666 MHz	8x	333 MHz QDR	2x 32+32 / 1x 6144 KB	1.3625 V	MMX SSE SSE2 SSE3 SSSE3 SSE4.1 NX EM64T VT	TM2 C1E EIST	65 W	72.4°C	410 Mio.	Wolfdale	C0	45 nm
Core 2 Duo E7300	2x 2666 MHz	10x	266 MHz QDR	2x 32+32 / 1x 3072 KB		MMX SSE SSE2 SSE3 SSSE3 SSE4.1 NX EM64T VT	TM2 C1E EIST	65 W			Wolfdale	M0	45 nm
Core 2 Duo E7200	2x 2533 MHz	9.5x	266 MHz QDR	2x 32+32 / 1x 3072 KB	1.3625 V	MMX SSE SSE2 SSE3 SSSE3 SSE4.1 NX EM64T VT	TM2 C1E EIST	65 W	74.1°C		Wolfdale	M0	45 nm
Core 2 Extreme QX6850	4x 3000 MHz	9x (open)	333 MHz QDR	4x 32+32 / 2x 4096 KB		MMX SSE SSE2 SSE3 SSSE3 NX EM64T VT	TM2 C1E EIST	130 W	64.5°C	582 Mio.	Kentsfield XE	G0	65 nm
Core 2 Extreme QX6800	4x 2933 MHz	11x (open)	266 MHz QDR	4x 32+32 / 2x 4096 KB	1.37 V	MMX SSE SSE2 SSE3 SSSE3 NX EM64T VT	TM2 C1E EIST	130 W	54.8°C	582 Mio.	Kentsfield XE	B3 G0	65 nm
Core 2 Extreme QX6700	4x 2666 MHz	10x (open)	266 MHz QDR	4x 32+32 / 2x 4096 KB	1.37 V	MMX SSE SSE2 SSE3 SSSE3 NX EM64T VT	TM2 C1E EIST	130 W	65°C	582 Mio.	Kentsfield XE	B3	65 nm
Core 2 Quad Q6700	4x 2666 MHz	10x	266 MHz QDR	4x 32+32 / 2x 4096 KB	1.37 V	MMX SSE SSE2 SSE3 SSSE3 NX EM64T VT	TM2 C1E EIST	95 W	71°C	582 Mio.	Kentsfield	G0	65 nm
Core 2 Quad Q6600	4x 2400 MHz	9x	266 MHz QDR	4x 32+32 / 2x 4096 KB	1.37 V	MMX SSE SSE2 SSE3 SSSE3 NX EM64T VT	TM2 C1E EIST	105 W	62.2°C	582 Mio.	Kentsfield	B3	65 nm
Core 2 Quad Q6600	4x 2400 MHz	9x	266 MHz QDR	4x 32+32 / 2x 4096 KB	1.37 V	MMX SSE SSE2 SSE3 SSSE3 NX EM64T VT	TM2 C1E EIST	95 W	71°C	582 Mio.	Kentsfield	G0	65 nm
Core 2 Extreme X6800	2x 2933 MHz	11x (open)	266 MHz QDR	2x 32+32 / 1x 4096 KB	1.35 V	MMX SSE SSE2 SSE3 SSSE3 NX EM64T VT	TM2 C1E EIST	75 W	60.4°C	291 Mio.	Conroe XE	B0 B1 B2	65 nm
Core 2 Duo E6850	2x 3000 MHz	9x	333 MHz QDR	4x 32+32 / 2x 4096 KB	1.5 V	MMX SSE SSE2 SSE3 SSSE3 NX EM64T VT	TM2 C1E EIST	65 W	72°C	291 Mio.	Conroe	G0	65 nm
Core 2 Duo E6750	2x 2666 MHz	8x	333 MHz QDR	4x 32+32 / 2x 4096 KB	1.5 V	MMX SSE SSE2 SSE3 SSSE3 NX EM64T VT	TM2 C1E EIST	65 W	72°C	291 Mio.	Conroe	G0	65 nm
Core 2 Duo E6550	2x 2333 MHz	7x	333 MHz QDR	4x 32+32 / 2x 4096 KB	1.5 V	MMX SSE SSE2 SSE3 SSSE3 NX EM64T VT	TM2 C1E EIST	65 W	72°C	291 Mio.	Conroe	G0	65 nm
Core 2 Duo E6540	2x 2333 MHz	7x	333 MHz QDR	4x 32+32 / 2x 4096 KB	1.5 V	MMX SSE SSE2 SSE3 SSSE3 NX EM64T VT	TM2 C1E EIST	65 W	72°C	291 Mio.	Conroe	G0	65 nm
Core 2 Duo E6700	2x 2667 MHz	10x	266 MHz QDR	2x 32+32 / 1x4096 KB	1.35 V	MMX SSE SSE2 SSE3 SSSE3 NX EM64T VT	TM2 C1E EIST	65 W	60.1°C	291 Mio.	Conroe	B0 B1 B2	65 nm
Core 2 Duo E6600	2x 2400 MHz	9x	266 MHz QDR	2x 32+32 / 1x4096 KB	1.32 V	MMX SSE SSE2 SSE3 SSSE3 NX EM64T VT	TM2 C1E EIST	65 W	60.1°C	291 Mio.	Conroe	B0 B1 B2	65 nm
Core 2 Duo E6400	2x 2133 MHz	8x	266 MHz QDR	2x 32+32 / 1x2048 KB	1.35 V	MMX SSE SSE2 SSE3 SSSE3 NX EM64T VT	TM2 C1E EIST	65 W	61.4°C	291 Mio.	Conroe-2048	B2	65 nm
Core 2 Duo E6300	2x 1866 MHz	7x	266 MHz QDR	2x 32+32 / 1x2048 KB	1.35 V	MMX SSE SSE2 SSE3 SSSE3 NX EM64T VT	TM2 C1E EIST	65 W	61.4°C	294 Mio.	Conroe-2048	B2	65 nm
Core 2 Duo E6420	2x 2133 MHz	8x	266 MHz QDR	2x 32+32 / 1x4096 KB		MMX SSE SSE2 SSE3 SSSE3 NX EM64T VT	TM2 C1E EIST	65 W	60.1°C	291 Mio.	Conroe	B2	65 nm
Core 2 Duo E6320	2x 1866 MHz	7x	266 MHz QDR	2x 32+32 / 1x4096 KB		MMX SSE SSE2 SSE3 SSSE3 NX EM64T VT	TM2 C1E EIST	65 W	60.1°C	291 Mio.	Conroe	B2	65 nm
Core 2 Duo E6400	2x 2133 MHz	8x	266 MHz QDR	2x 32+32 / 1x2048 KB	1.32 V	MMX SSE SSE2 SSE3 SSSE3 NX EM64T VT	TM2 C1E EIST	65 W	61.4°C	167 Mio.	Allendale	B0 B1 B2 L2	65 nm
Core 2 Duo E6300	2x 1866 MHz	7x	266 MHz QDR	2x 32+32 / 1x2048 KB	1.32 V	MMX SSE SSE2 SSE3 SSSE3 NX EM64T VT	TM2 C1E EIST	65 W	61.4°C	167 Mio.	Allendale	B1 B2 L2	65 nm
Core 2 Duo E4700	2x 2600 MHz	13x	200 MHz QDR	2x 32+32 / 1x2048 KB	1.312 V	MMX SSE SSE2 SSE3 SSSE3 NX EM64T	TM2 C1E EIST	65 W	73.3°C	167 Mio.	Allendale	G0	65 nm
Core 2 Duo E4600	2x 2400 MHz	12x	200 MHz QDR	2x 32+32 / 1x2048 KB	1.312 V	MMX SSE SSE2 SSE3 SSSE3 NX EM64T	TM2 C1E EIST	65 W	73.3°C	167 Mio.	Allendale	M0	65 nm
Core 2 Duo E4500	2x 2200 MHz	11x	200 MHz QDR	2x 32+32 / 1x2048 KB		MMX SSE SSE2 SSE3 SSSE3 NX EM64T	TM2 C1E EIST	65 W	73.3°C	167 Mio.	Allendale	L2 M0	65 nm
Core 2 Duo E4400	2x 2000 MHz	10x	200 MHz QDR	2x 32+32 / 1x2048 KB	1.31 V	MMX SSE SSE2 SSE3 SSSE3 NX EM64T	TM2 C1E EIST	65 W	61.4°C	167 Mio.	Allendale	L2	65 nm
Core 2 Duo E4300	2x 1800 MHz	9x	200 MHz QDR	2x 32+32 / 1x2048 KB	1.32 V	MMX SSE SSE2 SSE3 SSSE3 NX EM64T	TM2 C1E EIST	65 W	61.4°C	167 Mio.	Allendale	L2	65 nm
Pentium Dual Core E2220	2x 2400 MHz	12x	200 MHz QDR	2x 32+32 / 1x1024 KB	1.31 V	MMX SSE SSE2 SSE3 SSSE3 NX EM64T	TM2 C1E EIST	65 W			Conroe-L	M0	65 nm
Pentium Dual Core E2200	2x 2200 MHz	11x	200 MHz QDR	2x 32+32 / 1x1024 KB	1.31 V	MMX SSE SSE2 SSE3 SSSE3 NX EM64T	TM2 C1E EIST	65 W			Conroe-L	M0	65 nm
Pentium Dual Core E2180	2x 2000 MHz	10x	200 MHz QDR	2x 32+32 / 1x1024 KB	1.31 V	MMX SSE SSE2 SSE3 SSSE3 NX EM64T	TM2 C1E EIST	65 W			Conroe-L	M0	65 nm
Pentium Dual Core E2160	2x 1800 MHz	9x	200 MHz QDR	2x 32+32 / 1x1024 KB	1.31 V	MMX SSE SSE2 SSE3 SSSE3 NX EM64T	TM2 C1E EIST	65 W	61.4°C		Conroe-L	L2 G0 M0	65 nm
Pentium Dual Core E2140	2x 1600 MHz	8x	200 MHz QDR	2x 32+32 / 1x1024 KB	1.31 V	MMX SSE SSE2 SSE3 SSSE3 NX EM64T	TM2 C1E EIST	65 W	61.4°C		Conroe-L	L2 G0 M0	65 nm
Celeron E1400	2x 2000 MHz	10x	200 MHz QDR	2x 32+32 / 512 KB	1.312 V	MMX SSE SSE2 SSE3 SSSE3 NX EM64T	TM2	65 W	73.3°C		Conroe-L	M0	65 nm
Celeron E1200	2x 1600 MHz	8x	200 MHz QDR	2x 32+32 / 512 KB	1.312 V	MMX SSE SSE2 SSE3 SSSE3 NX EM64T	TM2	65 W	73.3°C		Conroe-L	M0	65 nm
Celeron 440	2000 MHz	10x	200 MHz QDR	32+32 / 512 KB	1.3375 V	MMX SSE SSE2 SSE3 SSSE3 NX EM64T	TM2	35 W	60.4°C		Conroe-L	A1	65 nm
Celeron 430	1800 MHz	9x	200 MHz QDR	32+32 / 512 KB	1.3375 V	MMX SSE SSE2 SSE3 SSSE3 NX EM64T	TM2	35 W	60.4°C		Conroe-L	A1	65 nm
Celeron 420	1600 MHz	8x	200 MHz QDR	32+32 / 512 KB	1.3375 V	MMX SSE SSE2 SSE3 SSSE3 NX EM64T	TM2	35 W	60.4°C		Conroe-L	A1	65 nm
Pentium EE 965	2x 3724 MHz	14x	266 MHz QDR	12KμOps+16/2048 KB	1.30 V	MMX SSE SSE2 SSE3 NX EM64T VT HT	TM1 C1E	130 W	68.6°C	376 Mio.	Presler	C1	65 nm
Pentium EE 955	2x 3466 MHz	13x	266 MHz QDR	12KμOps+16/2048 KB	1.30 V	MMX SSE SSE2 SSE3 NX EM64T VT HT	TM1	130 W	68.6°C	376 Mio.	Presler	B1	65 nm
Pentium D 960	2x 3600 MHz	18x	200 MHz QDR	12KμOps+16/2048 KB	1.30 V	MMX SSE SSE2 SSE3 NX EM64T VT	TM1 C1E	130 W	86.6°C	376 Mio.	Presler	C1	65 nm
Pentium D 960	2x 3600 MHz	18x	200 MHz QDR	12KμOps+16/2048 KB	1.31 V	MMX SSE SSE2 SSE3 NX EM64T VT	TM1 C1E	95 W	63.4°C	376 Mio.	Presler	D0	65 nm
Pentium D 950	2x 3400 MHz	17x	200 MHz QDR	12KμOps+16/2048 KB	1.40 V	MMX SSE SSE2 SSE3 NX EM64T VT	TM1 C1E	130 V	86.6°C	376 Mio.	Presler	B1	65 nm
Pentium D 950	2x 3400 MHz	17x	200 MHz QDR	12KμOps+16/2048 KB	1.31 V	MMX SSE SSE2 SSE3 NX EM64T VT	TM1 C1E	95 W	63.4°C	376 Mio.	Presler	C1 D0	65 nm
Pentium D 945	2x 3200 MHz	16x	200 MHz QDR	12KμOps+16/2048 KB	1.32 V	MMX SSE SSE2 SSE3 NX EM64T	TM1 C1E	95 W	63.4°C	376 Mio.	Presler	C1 D0	65 nm
Pentium D 940	2x 3200 MHz	16x	200 MHz QDR	12KμOps+16/2048 KB	1.40 V	MMX SSE SSE2 SSE3 NX EM64T VT	TM1 C1E	130 V	86.6°C	376 Mio.	Presler	B1	65 nm
Pentium D 940	2x 3200 MHz	16x	200 MHz QDR	12KμOps+16/2048 KB	1.31 V	MMX SSE SSE2 SSE3 NX EM64T VT	TM1 C1E	95 W	63.4°C	376 Mio.	Presler	C1	65 nm
Pentium D 935	2x 3000 MHz	15x	200 MHz QDR	12KμOps+16/2048 KB	1.32 V	MMX SSE SSE2 SSE3 NX EM64T	TM1 C1E	95 W	63.4°C	376 Mio.	Presler	D0	65 nm
Pentium D 930	2x 3000 MHz	15x	200 MHz QDR	12KμOps+16/2048 KB	1.40 V	MMX SSE SSE2 SSE3 NX EM64T VT	TM1 C1E	95 W	63.4°C	376 Mio.	Presler	B1 C1	65 nm
Pentium D 925	2x 2800 MHz	14x	200 MHz QDR	12KμOps+16/2048 KB	1.31 V	MMX SSE SSE2 SSE3 NX EM64T	TM1 C1E	95 W	63.4°C	376 Mio.	Presler	C1 D0	65 nm
Pentium D 920	2x 2800 MHz	14x	200 MHz QDR	12KμOps+16/2048 KB	1.40 V	MMX SSE SSE2 SSE3 NX EM64T VT	TM1 C1E	95 W	63.4°C	376 Mio.	Presler	B1	65 nm
Pentium D 915	2x 2800 MHz	14x	200 MHz QDR	12KμOps+16/2048 KB	1.31 V	MMX SSE SSE2 SSE3 NX EM64T	TM1 C1E	95 W	63.4°C	376 Mio.	Presler	C1 D0	65 nm
Pentium 4 "E" 661	3600 MHz	18x	200 MHz QDR	12KμOps+16/2048 KB	1.32 V	MMX SSE SSE2 SSE3 NX EM64T HT	TM1 C1E	86 W	69.2°C	188 Mio.	Cedar Mill	B1 C1	65 nm
Pentium 4 "E" 651	3400 MHz	17x	200 MHz QDR	12KμOps+16/2048 KB	1.32 V	MMX SSE SSE2 SSE3 NX EM64T HT	TM1 C1E	86 W	69.2°C	188 Mio.	Cedar Mill	B1 C1	65 nm
Pentium 4 "E" 651	3400 MHz	17x	200 MHz QDR	12KμOps+16/2048 KB	1.32 V	MMX SSE SSE2 SSE3 NX EM64T HT	TM1 C1E	65 W	64.4°C	188 Mio.	Cedar Mill	D0	65 nm
Pentium 4 "E" 641	3200 MHz	16x	200 MHz QDR	12KμOps+16/2048 KB	1.32 V	MMX SSE SSE2 SSE3 NX EM64T HT	TM1 C1E	86 W	69.2°C	188 Mio.	Cedar Mill	B1 C1	65 nm
Pentium 4 "E" 641	3200 MHz	16x	200 MHz QDR	12KμOps+16/2048 KB	1.32 V	MMX SSE SSE2 SSE3 NX EM64T HT	TM1 C1E	65 W	64.4°C	188 Mio.	Cedar Mill	D0	65 nm
Pentium 4 "E" 631	3000 MHz	15x	200 MHz QDR	12KμOps+16/2048 KB	1.32 V	MMX SSE SSE2 SSE3 NX EM64T HT	TM1 C1E	86 W	69.2°C	188 Mio.	Cedar Mill	B1 C1	65 nm
Pentium 4 "E" 631	3000 MHz	15x	200 MHz QDR	12KμOps+16/2048 KB	1.32 V	MMX SSE SSE2 SSE3 NX EM64T HT	TM1 C1E	65 W	69.2°C	188 Mio.	Cedar Mill	D0	65 nm

CPU Queen

CPU	CPU Clock	Motherboard	Chipset	Memory	CL-RCD-RP-RAS	Score
4x Core i7 Extreme 965 HT	3333 MHz	Asus P6T Deluxe	X58	Triple DDR3-1333	9-9-9-24 CR1	30786
8x Xeon E5462	2800 MHz	Intel S5400SF	i5400	Quad DDR2-640FB	5-5-5-15	30472
4x Core 2 Extreme QX9650	3000 MHz	Gigabyte GA-EP35C-DS3R	P35	Dual DDR3-1066	8-8-8-20 CR2	21421
8x Xeon L5320	1866 MHz	Intel S5000VCL	i5000V	Dual DDR2-533FB	4-4-4-12	20452
4x Core 2 Extreme QX6700	2666 MHz	Intel D975XBX2	i975X	Dual DDR2-667	5-5-5-15	19166
4x Phenom II X4 Black 940	3000 MHz	Asus M3N78-EM	GeForce8300 Int.	Ganged Dual DDR2-800	5-5-5-18 CR2	18636
4x Xeon 5140	2333 MHz	Intel S5000VSA	i5000V	Dual DDR2-667FB	5-5-5-15	16729
8x Opteron HE 2344	1700 MHz	Tyan Thunder n3600R	nForcePro-3600	Unganged Dual DDR2-667R	5-5-5-15 CR1	16146
4x Phenom X4 9500	2200 MHz	Asus M3A	AMD770	Ganged Dual DDR2-800	5-5-5-18 CR2	13693
2x Core 2 Duo E6700	2666 MHz	Abit AB9	P965	Dual DDR2-800	5-5-5-18 CR2	11406
2x Athlon64 X2 Black 6400+	3200 MHz	MSI K9N SLI Platinum	nForce570SLI	Dual DDR2-800	4-4-4-11 CR1	11169
2x Core 2 Duo P8400	2266 MHz	MSI MegaBook PR201	GM45 Int.	Dual DDR2-667	5-5-5-15	9578
2x Pentium T3400	**2166 MHz**	**Toshiba Satellite L305**	**GL40 Int.**	**Dual DDR2-667**	**5-5-5-13**	**9145**
2x Core Duo T2500	2000 MHz	Asus N4L-VM DH	i945GT Int.	Dual DDR2-667	5-5-5-15	7793
2x Core 2 Duo T5600	1833 MHz	Asus F3000Jc Notebook	i945PM	Dual DDR2-667	5-5-5-15	7717
2x Athlon64 X2 4000+	2100 MHz	ASRock ALiveNF7G-HDready	nForce7050-630a Int.	Dual DDR2-700	5-5-5-18 CR2	7280
2x Pentium EE 955 HT	3466 MHz	Intel D955XBK	i955X	Dual DDR2-667	4-4-4-11	7098
2x Xeon	3066 MHz	Asus PCH-DL	i875P + PAT	Dual DDR333	2-2-2-5	6188
2x Opteron 240	1400 MHz	MSI K8D Master3-133 FS	AMD8100	Dual DDR400R	3-4-4-8 CR1	4863
2x PIII-S	1266 MHz	MSI Pro266TD Master-LR	ApolloPro266TD	DDR266 SDRAM	2-3-3-6 CR2	4857
P4EE HT	3733 MHz	Intel SE7230NH1LX	iE7230	Dual DDR2-667	5-5-5-15	4210
Opteron 248	2200 MHz	MSI K8T Master1-FAR	K8T800	Dual DDR266R	2-3-3-6 CR1	3851
Atom 230	1600 MHz	Intel D945GCLF	i945GC	DDR2-533 SDRAM	4-4-4-12	3779

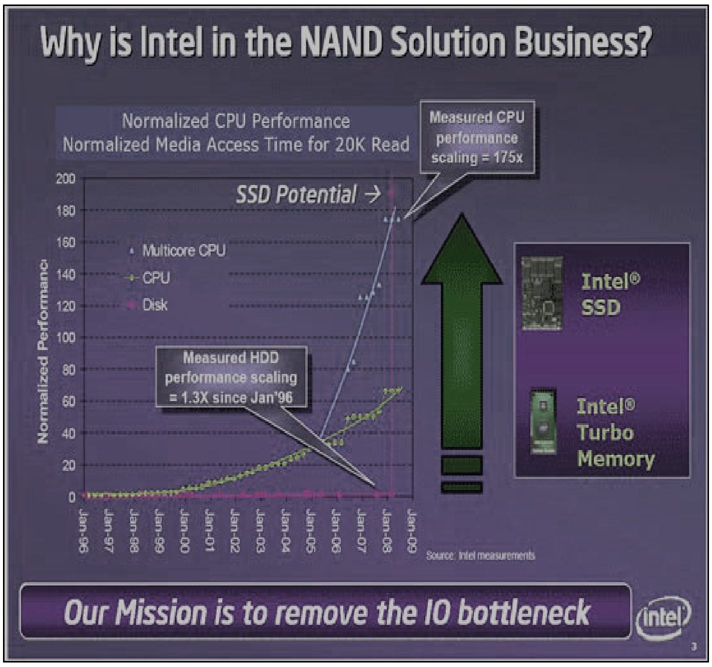
CPU AES

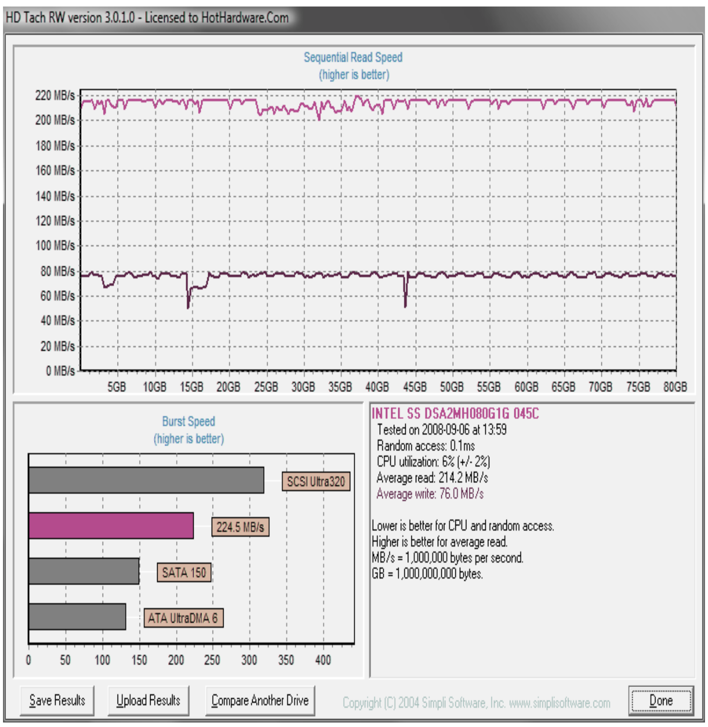
CPU	CPU Clock	Motherboard	Chipset	Memory	CL-RCD-RP-RAS	Score
8x Xeon E5462	2800 MHz	Intel S5400SF	i5400	Quad DDR2-640FB	5-5-5-15	41625
8x Xeon L5320	1866 MHz	Intel S5000VCL	i5000V	Dual DDR2-533FB	4-4-4-12	27698
4x Core i7 Extreme 965 HT	3333 MHz	Asus P6T Deluxe	X58	Triple DDR3-1333	9-9-9-24 CR1	26703
8x Opteron HE 2344	1700 MHz	Tyan Thunder n3600R	nForcePro-3600	Unganged Dual DDR2-667R	5-5-5-15 CR1	22599
4x Core 2 Extreme QX9650	3000 MHz	Gigabyte GA-EP35C-DS3R	P35	Dual DDR3-1066	8-8-8-20 CR2	22435
4x Phenom II X4 Black 940	3000 MHz	Asus M3N78-EM	GeForce8300 Int.	Ganged Dual DDR2-800	5-5-5-18 CR2	21658
4x Core 2 Extreme QX6700	2666 MHz	Intel D975XBX2	i975X	Dual DDR2-667	5-5-5-15	19896
C7	1500 MHz	VIA EPIA EN	CN700 Int.	DDR2-533 SDRAM	4-4-4-12 CR2	17358
4x Xeon 5140	2333 MHz	Intel S5000VSA	i5000V	Dual DDR2-667FB	5-5-5-15	17320
4x Phenom X4 9500	2200 MHz	Asus M3A	AMD770	Ganged Dual DDR2-800	5-5-5-18 CR2	14802
2x Core 2 Duo E6700	2666 MHz	Abit AB9	P965	Dual DDR2-800	5-5-5-18 CR2	9969
2x Pentium EE 955 HT	3466 MHz	Intel D955XBK	i955X	Dual DDR2-667	4-4-4-11	8970
2x Core 2 Duo P8400	2266 MHz	MSI MegaBook PR201	GM45 Int.	Dual DDR2-667	5-5-5-15	8443
2x Athlon64 X2 Black 6400+	3200 MHz	MSI K9N SLI Platinum	nForce570SLI	Dual DDR2-800	4-4-4-11 CR1	8339
2x Pentium T3400	**2166 MHz**	**Toshiba Satellite L305**	**GL40 Int.**	**Dual DDR2-667**	**5-5-5-13**	**7966**
2x Core Duo T2500	2000 MHz	Asus N4L-VM DH	i945GT Int.	Dual DDR2-667	5-5-5-15	7109

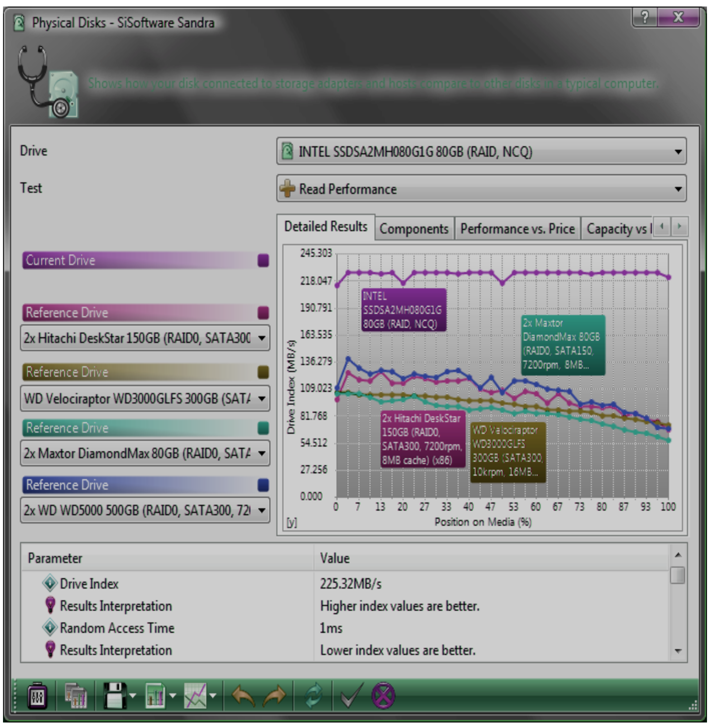
CPU ZLib

CPU	CPU Clock	Motherboard	Chipset	Memory	CL-RCD-RP-RAS	Score
8x Xeon E5462	2800 MHz	Intel S5400SF	i5400	Quad DDR2-640FB	5-5-5-15	139481 KB/s
4x Core i7 Extreme 965 HT	3333 MHz	Asus P6T Deluxe	X58	Triple DDR3-1333	9-9-9-24 CR1	112330 KB/s
8x Xeon L5320	1866 MHz	Intel S5000VCL	i5000V	Dual DDR2-533FB	4-4-4-12	95887 KB/s
8x Opteron HE 2344	1700 MHz	Tyan Thunder n3600R	nForcePro-3600	Unganged Dual DDR2-667R	5-5-5-15 CR1	88845 KB/s
4x Phenom II X4 Black 940	3000 MHz	Asus M3N78-EM	GeForce8300 Int.	Ganged Dual DDR2-800	5-5-5-18 CR2	80081 KB/s
4x Core 2 Extreme QX9650	3000 MHz	Gigabyte GA-EP35C-DS3R	P35	Dual DDR3-1066	8-8-8-20 CR2	77167 KB/s
4x Core 2 Extreme QX6700	2666 MHz	Intel D975XBX2	i975X	Dual DDR2-667	5-5-5-15	69756 KB/s
4x Xeon 5140	2333 MHz	Intel S5000VSA	i5000V	Dual DDR2-667FB	5-5-5-15	60995 KB/s
4x Phenom X4 9500	2200 MHz	Asus M3A	AMD770	Ganged Dual DDR2-800	5-5-5-18 CR2	58396 KB/s
2x Athlon64 X2 Black 6400+	3200 MHz	MSI K9N SLI Platinum	nForce570SLI	Dual DDR2-800	4-4-4-11 CR1	38059 KB/s
2x Core 2 Duo E6700	2666 MHz	Abit AB9	P965	Dual DDR2-800	5-5-5-18 CR2	35355 KB/s
2x Pentium EE 955 HT	3466 MHz	Intel D955XBK	i955X	Dual DDR2-667	4-4-4-11	29844 KB/s
2x Core 2 Duo P8400	2266 MHz	MSI MegaBook PR201	GM45 Int.	Dual DDR2-667	5-5-5-15	29223 KB/s
2x Pentium T3400	**2166 MHz**	**Toshiba Satellite L305**	**GL40 Int.**	**Dual DDR2-667**	**5-5-5-13**	**28519 KB/s**
2x Xeon	3066 MHz	Asus PCH-DL	i875P + PAT	Dual DDR333	2-2-2-5	25040 KB/s
2x Core 2 Duo T5600	1833 MHz	Asus F3000Jc Notebook	i945PM	Dual DDR2-667	5-5-5-15	23969 KB/s
2x Athlon64 X2 4000+	2100 MHz	ASRock ALiveNF7G-HDready	nForce7050-630a Int.	Dual DDR2-700	5-5-5-18 CR2	23852 KB/s

Intel claims 230MB/s read, (benchmark tools say 214MB/s read) - already on market:



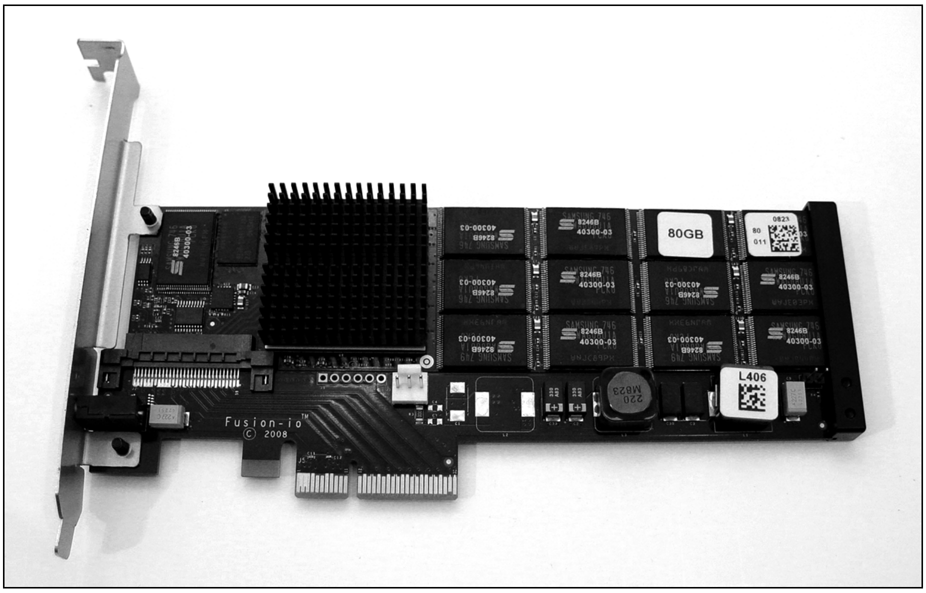


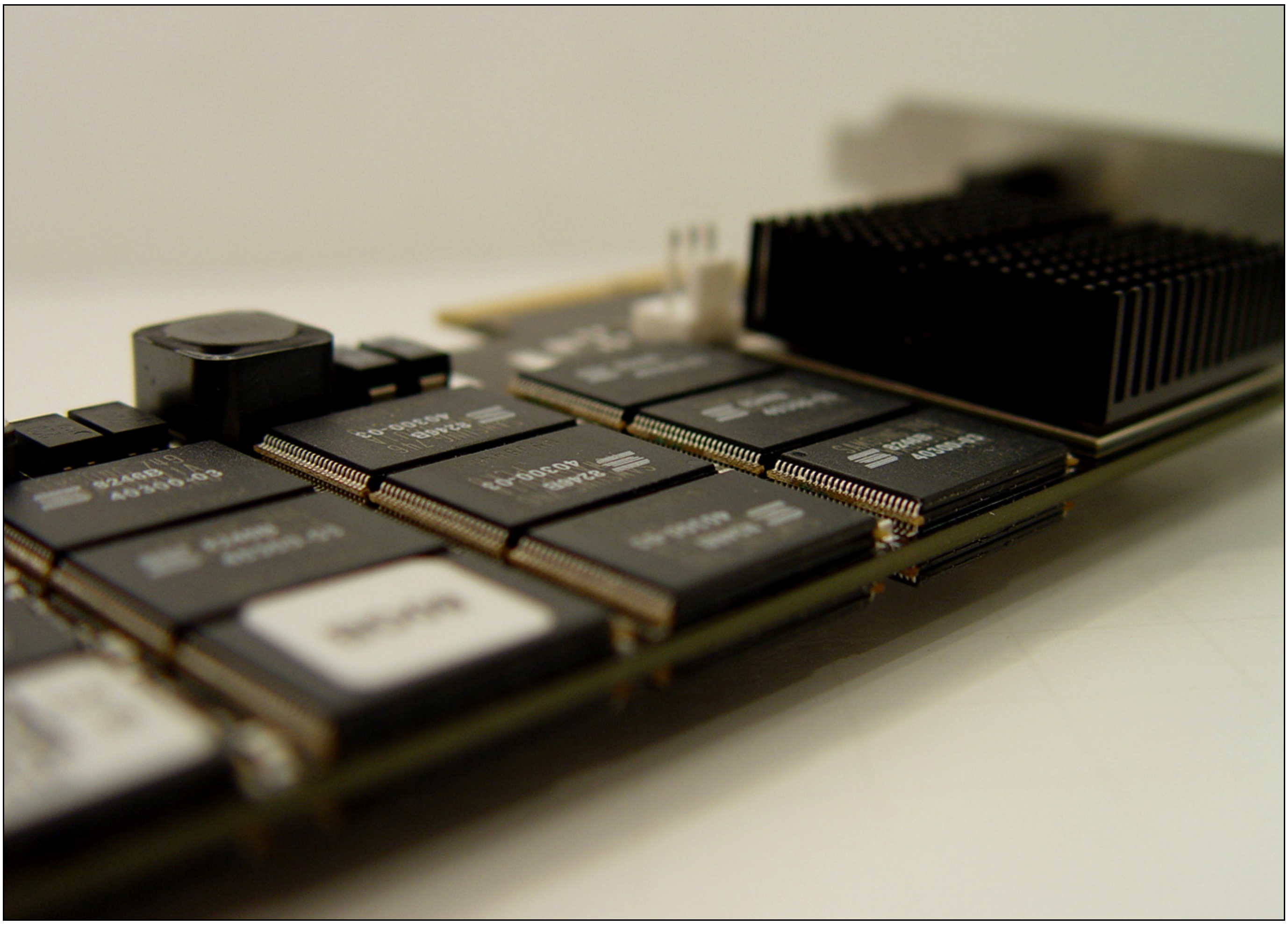


***More SSD Exuberance** – wow-things or as they said ‘MIND BLOWING’:*

***Fusion ioDrive** SPECIFICATIONS:*

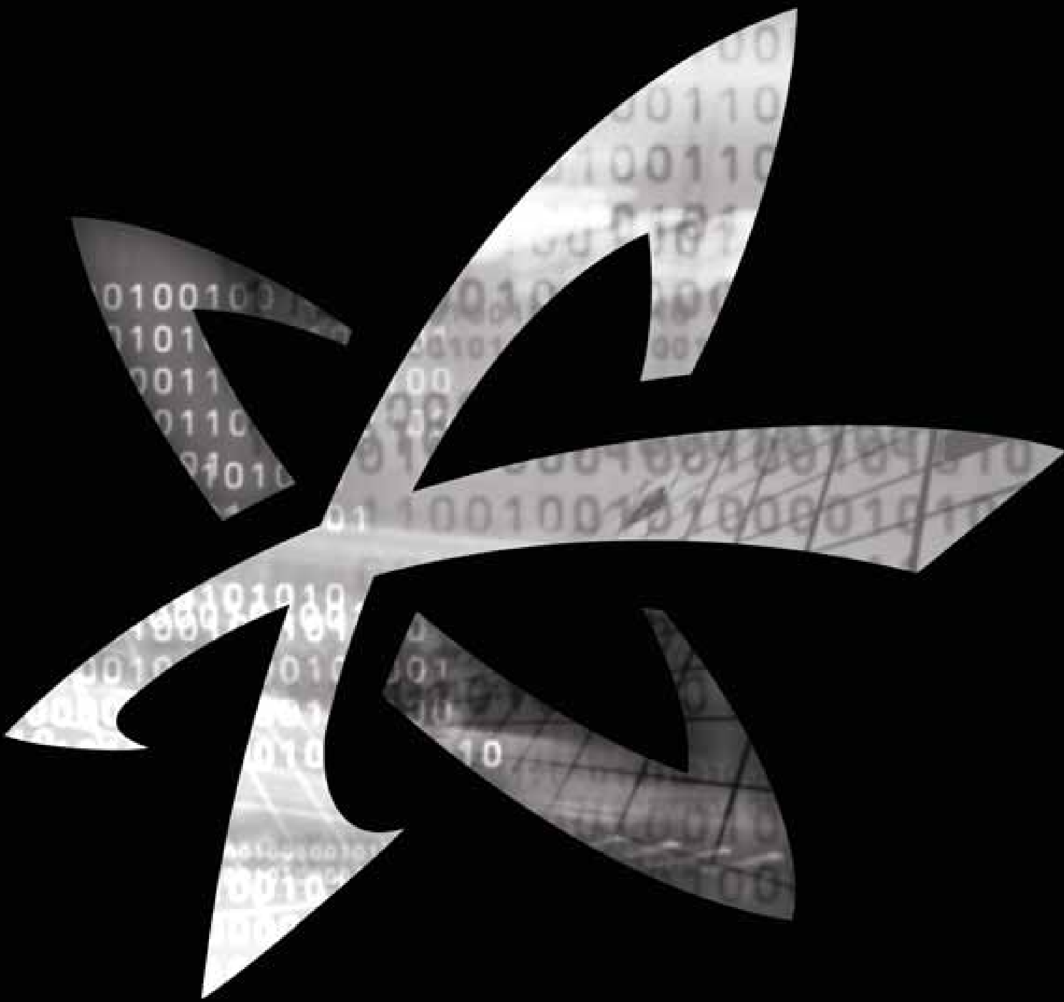
NAND Type: Single Level Cell (SLC)
Read Bandwidth: 700 MB/s (random 16K)
Access Latency: 50µs
Bus Interface: PCI-Express x4
Operating Systems: Microsoft 64-Bit Windows(64-Bit Windows XP, Vista, Server 2003 & 2008)





* * *

With the **ioDrive Duo**, it is now possible for application, database and system administrators to get previously unheard-of levels of performance, protection and capacity utilization from a single server. Performance for multiple **ioDrive Duos** scales linearly, allowing any enterprise to scale performance to **six gigabytes per-second** (Gbytes/sec) of read bandwidth and over 500,000 read IOPS by using just four **ioDrive Duos.**



Fusion-io's Solid State Storage – A New Standard for Enterprise-Class Reliability

FUSION-IO

Fusion-io's Solid State Storage – A New Standard for Enterprise-Class Reliability

Fusion-io offers solid state storage solutions based on NAND flash that provide a level of integrity and availability for mission-critical data that exceeds today's solid state storage solutions and significantly surpasses that of enterprise-class rotating magnetic storage devices.

With throughput and seek times many times faster than the fastest disk arrays, it is little wonder that enterprise data centers have been keen to include NAND flash as part of their server infrastructure. The primary reason NAND flash has not been widely adopted in the computer industry is its reputation for unreliability. There is a long-standing view that NAND flash storage works well for non-mission-critical applications, such as media storage devices (where the occasional bit error generally translates into a slight audio hiss or a stray errant pixel in a video), but cannot be relied upon for applications where a bit error could crash an operating system or compromise the integrity of critical data.

System architects face a number of storage-related challenges and NAND flash technology presents its own set of unique problems. But Fusion-io has developed patent-pending techniques to create NAND flash-based storage with reliability equal to or exceeding that of disk-based storage. This paper describes several inventions and advancements Fusion-io has introduced to ensure data is not corrupted or lost. Additionally, this paper discusses the probability of catastrophic storage device failure and how Fusion-io's architecture ensures predicable, controlled management of early device failure, long-term device attrition and data changes due to external and data transport interference.

NAND Flash

Flash memory chips are a non-volatile storage medium (i.e., they can retain their information even in the absence of power). The most common types of flash chips are silicon-based NOR and NAND, named after the types of logic gates used in their design. NAND flash, introduced in 1989, has become the most commonly used type of flash chip, due to its quicker write speed. Flash memory continues to grow in popularity as its price steadily declines, its storage capacity increases, and its physical size continues to decrease.

In Fusion-io's storage devices, NAND flash chips are stacked several at a time (to increase density), operated in parallel (to increase throughput) and mounted on a printed circuit board (PCB) that plugs into a PCI-Express (PCIe) slot on the server or in the CPU. The flash media is integrated with the controller onto a single PCI-Express card.

NAND flash, as a storage medium, offers a number of benefits in comparison to rotating magnetic storage devices (aka HDD, Hard Disk Drives). NAND flash has no moving parts and is therefore significantly less prone to shock or movement disturbance. It is a high speed solution in both latency and throughput. Temperature and humidity resistance mean that it can operate in a number of different environments. Finally, NAND flash consumes significantly less power than rotating magnetic storage devices, particularly when you take into account secondary power requirements for device cooling.

However, NAND flash does introduce a number of potential failure points including:

- Media – Media failures can occur on the NAND flash chips themselves.
- Transport – Transport errors can occur anywhere along the path carrying data from the CPU through to the NAND flash chips.
- Management – There is a small chance that management problems can occur within the logic of the device itself. The code that controls the operation can contain technical problems that can result in data failures.
- External – External problems can affect any part of the process.
- Device Failure – Catastrophic hardware failure can also occur. This includes the possibility of internal short circuits and open circuits within the memory array itself.

Protecting the Data

Implementing a variety of design and architectural strategies for protecting data integrity, Fusion-io's NAND flash devices greatly exceed the reliability of rotating magnetic media storage devices, while providing performance that is orders of magnitude better. Fusion-io protects your data at every step, ensuring that nothing is lost or corrupted in transit or on the media.

Data Integrity

Data integrity means having a high degree of confidence that what you put into a storage system is exactly what you get out when you request that data and it is the most important function of a storage system. While being moved from a computer's RAM or CPU to the Fusion-io device, several proven industry-standard approaches are used to ensure data integrity. The CPU, chipset, and RAM use SECDED (Single Error Correct Double Error Detect) or chipkill (method for on-the-fly replacement of a failed chip) to ensure accuracy. Once data is written to the storage medium, it is again checked for accuracy.

When data is read from the storage medium, error correction techniques are again employed to ensure that the data being retrieved is correct. The device can correct a substantial portion of the data being read. NAND's reputation for unreliability is based on studies that show potential data loss without utilizing error correction – or less correction than that employed by the Fusion-io device. Using the methods described here, Fusion-io devices can produce results that exceed target error probability by about four times. Fusion-io's devices also use a patent-pending approach when writing data, which allows the data's path to be reconstructed from information generated during the write process.

Data Availability

Data availability means having a high degree of confidence that data stored will not be lost, either while in transition to the storage device or after it has been written to the media.

Fusion-io employs a wide variety of techniques to overcome some of the common problems associated with data availability in general, and also addresses some that are particular to NAND flash as a storage medium. Generally speaking, NAND flash is substantially more reliable than rotating magnetic media. It eliminates the chance of mechanical failure (the failure associated with moving parts). There is, however, a chance of bad chips and chip wear-out. Fusion-io mitigates this risk using a variety of approaches.

Fusion-io's redundant, patent-pending approach to writing data allows data to be rebuilt at a very high rate of speed, ensuring rapid data availability. Data is also regularly moved and checked for accuracy to ensure that it does not deteriorate on the flash chip. This also consolidates good data and reallocates space on the drive to ensure greater data availability. This system also spreads data evenly across the device, ensuring uniform wear across all chips.

Additionally, Fusion-io uses multiple error correction code (ECC) techniques to identify and correct faulty data. Using ECCs, the device controller can correct up to 11 missing or incorrect bits out of every 240 bytes. One of the biggest benefits of ECC routines is that it they allow the device to predict the likelihood of failure on individual chips. When a particular area of a chip has passed a set unreliability threshold, its data can be moved and that are will be taken out of service. The controller continues to identify and remove bad blocks, regions of chips or even entire chips so that ordinary wear-out does not cause catastrophic failure rather a very predictable wear-out.

Device Longevity

The majority of this paper has concentrated on NAND flash in an enterprise-class storage device, and how to leverage its strengths while overcoming its weaknesses. NAND flash, however, is only part of a Fusion-io's storage device. The flash chips reside on a PCIe adapter card that has a number of other parts as well, all of which are susceptible to failure. The life of a NAND flash storage device can be estimated by examining the failure rate of its component parts. Wear-out is generally a function of having lost enough storage cells that both capacity and reliability drop below acceptable thresholds. This can be assessed by evaluating and keeping a record of the amount of errors detected at each physical location.

NAND flash wears out at a predictable rate as described by the formulas below. Effective use of wear-leveling strategies employed by Fusion-io can significantly improve the life expectancy of its drives. Please note that the formulas are applied to both MLC and SLC NAND-based non-volatile memory technologies. Single-Level Cell (SLC) NAND and Multi-Level Cell (MLC) NAND offer capabilities that serve two very different types of applications – respectively, those requiring high performance at an attractive cost-per-bit and those seeking even higher performance over time, that are less cost-sensitive:

Average-lifetime = lifetime / read-write- ratio

TYPE / WRITE DUTY	AVERAGE ESTIMATED LIFETIME FORMULA
SLC flash @ 40% write duty	25 calendar years
MLC flash @ 20% write duty	10 calendar years
MLC flash @ 40% write duty	5 calendar years

Average estimated lifetime based on Fusion-io lab testing

The read/write ratio is difficult to predict, and will vary considerably from environment to environment. As a point of reference, the International Disk-drive Equipment and Materials Association (IDEMA), an industry trade group that publishes storage device standards, recommends a read/write ratio of 60%/40% for its server-class device reliability testing (IDEMA Standards, Document R3-98).

Flashback Protection

Enterprises have long sought to take advantage of the speed, size, low-power and high-performance of NAND Flash because of its potential to change the way they manage large amounts of active data. The primary objection to NAND flash has been the reliability of the medium. Fusion-io has eliminated this barrier by inventing a revolutionary self-healing technology, known as Flashback Protection, in our controllers that instantaneously restores, corrects and resurrects lost data in the flash-based storage sub-system. Flashback Protection is accomplished by collectively using advanced bit error correction, proactive data integrity monitoring of stored data and the recent addition of a dedicated chip to repair failed devices.

Fusion-io is the first and only company to bring RAID-class redundancy and reliability using Flashback Protection down to the card level. The Flashback Protection system allows users to diagnose and correct system errors. Fusion-io integrates dedicated NAND flash chips, which offer information that enables the detection of single bit errors. This technique eliminates data loss due to chip failures and extends the usable lifetime of the NAND flash-based storage device. The NAND flash chips on Fusion-io's products contain an innovative storage architecture that enable it to deliver the performance, and now the reliability, of a storage area network (SAN) at a fraction of the power, size and cost of traditional disk arrays.

Controlled Predictable Usage Versus Catastrophic Failure

Among the greatest reliability benefits of the Fusion-io storage device is its ability to:

- Restore and Protect data
- Monitor and predict media wear-out
- Correct bad data as necessary
- Take blocks out of service when their failure rate becomes unacceptable
- Replace bad chips on-the-fly
- Move the data to a known good location (and update corresponding mapping information)

Data stored on the Fusion-io medium is double protected using both ECCs and parity data on the redundant chip. The net effect is that wear-out of the device, instead of being catastrophic, is predictable and incremental.

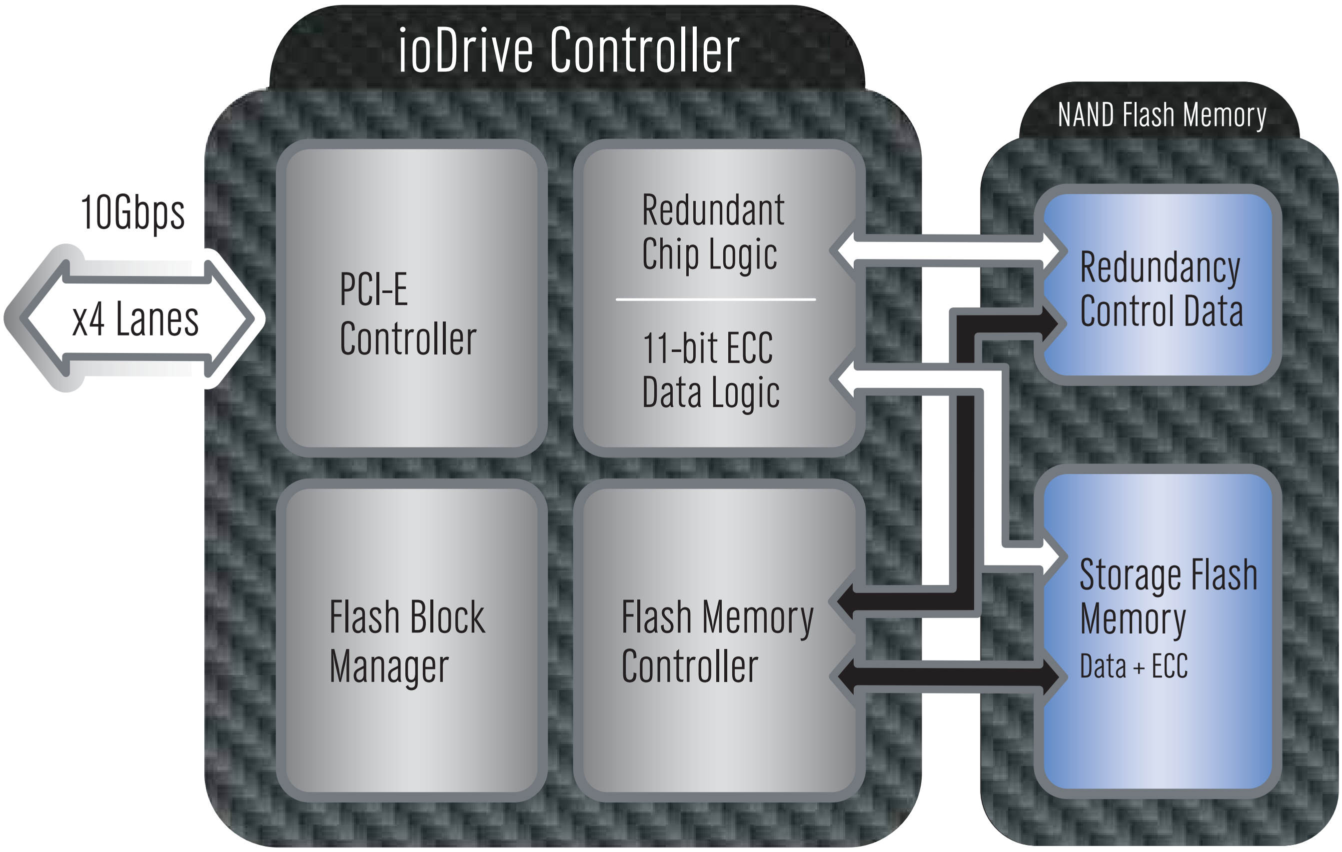
A Fusion-io device provides advanced warning prior to wear-out. Fusion-io supports today's monitoring management functions to measure and report on the device's status and usable life. In almost all cases, device upgrade is a smooth and predictable process, rather than an emergency situation.

Fusion-io protects your data at every stage of its path from your applications to the NAND flash storage medium, ensuring that nothing is lost or corrupted along the way or while the data is being stored. Data is checked multiple times, using several error detection methods. Once it reaches the storage medium, it is stored with robust error correction encoding that lets the flash device not only identify but correct bit errors. Fusion-io's data integrity design target is a 1 in 10^{30} probability of undetected bad data and a 1 in 10^{20} probability of uncorrectable data, as compared to a 1 in 10^{16} probability of undetected or uncorrectable errors for rotating magnetic storage devices.

Conclusion

Now with Fusion-io's comprehensive approach to data integrity, it is safe to exploit the exponential performance gains and many other benefits offered by NAND flash storage. The storage architecture pioneered by Fusion-io ensures predictable, controlled mitigation of early device failure, long-term device attrition and data changes due to external and data transport interference—issues that have up to now limited the adoption of NAND flash-based storage at the enterprise level. Fusion-io's NAND flash devices exceed the reliability of rotating magnetic media storage devices while providing an order of magnitude performance improvement.

Flash Back Block Diagram



FUSION-IO

Robert Brumfield
Fusion Public Relations
212.651.4215
robert.brumfield@fusionpr.com

Fusion-io Announces the ioDrive Duo—The World's Fastest and Most Innovative SSD

PCI Express, server-based solid-state storage offering sets a new standard for enterprise application-centric storage, with up to 640 gigabytes of capacity and 1.5 gigabytes per-second of sustained throughput

SALT LAKE CITY - March 11, 2009 - Fusion-io, the leader in solid-state architecture and high-performance I/O solutions, today announced the ioDrive Duo, which doubles the slot capacity of Fusion-io's successful PCI Express-based ioDrive storage solution. The new ioDrive Duo is the market's fastest and most innovative server-based solid-state storage solution.

With the ioDrive Duo, it is now possible for application, database and system administrators to get previously unheard-of levels of performance, protection and capacity utilization from a single server. Performance for multiple ioDrive Duos scales linearly, allowing any enterprise to scale performance to six gigabytes per-second (Gbytes/sec) of read bandwidth and over 500,000 read IOPS by using just four ioDrive Duos.

"Many database and system administrators are finding that SANs are too expensive and don't meet performance, protection and capacity utilization expectations," said David Flynn, CTO of Fusion-io. "This is why more and more application vendors are moving toward application-centric solid-state storage. The ioDrive Duo offers the enterprise the advantages of application-centric storage without application-specific programming."

ioDrive Duo Product Details

The following specifications describe the physical and performance characteristics of the ioDrive Duo.

Performance

Based on PCI Express x8 or PCI Express 2.0 x4 standards, which can sustain up to 20 gigabits per-second of raw throughput, the ioDrive Duo has more than enough bandwidth to obtain industry-leading performance from a single card. The ioDrive Duo can easily sustain 1.5 Gbytes/sec of read bandwidth and nearly 200,000 read IOPS. Its performance metrics are as follows:

- Sustained read bandwidth: 1500 MB/sec (32k packet size)
- Sustained write bandwidth: 1400 MB/sec (32k packet size)
- Read IOPS: 186,000 (4k packet size)
- Write IOPS: 167,000 (4k packet size)
- Latency < 50 µsec

Reliability

The ioDrive Duo offers unmatched solid-state protection for data integrity and reliability with triple redundancy for a single storage component.

- Multi-bit error detection and correction
- Patent-pending Flashback protection, offering chip-level N+1 redundancy and on-board self-healing so that no servicing is required
- Optional RAID-1 mirroring between two ioMemory modules on the same ioDrive Duo, offering complete redundancy on a single PCIe card

Capacity

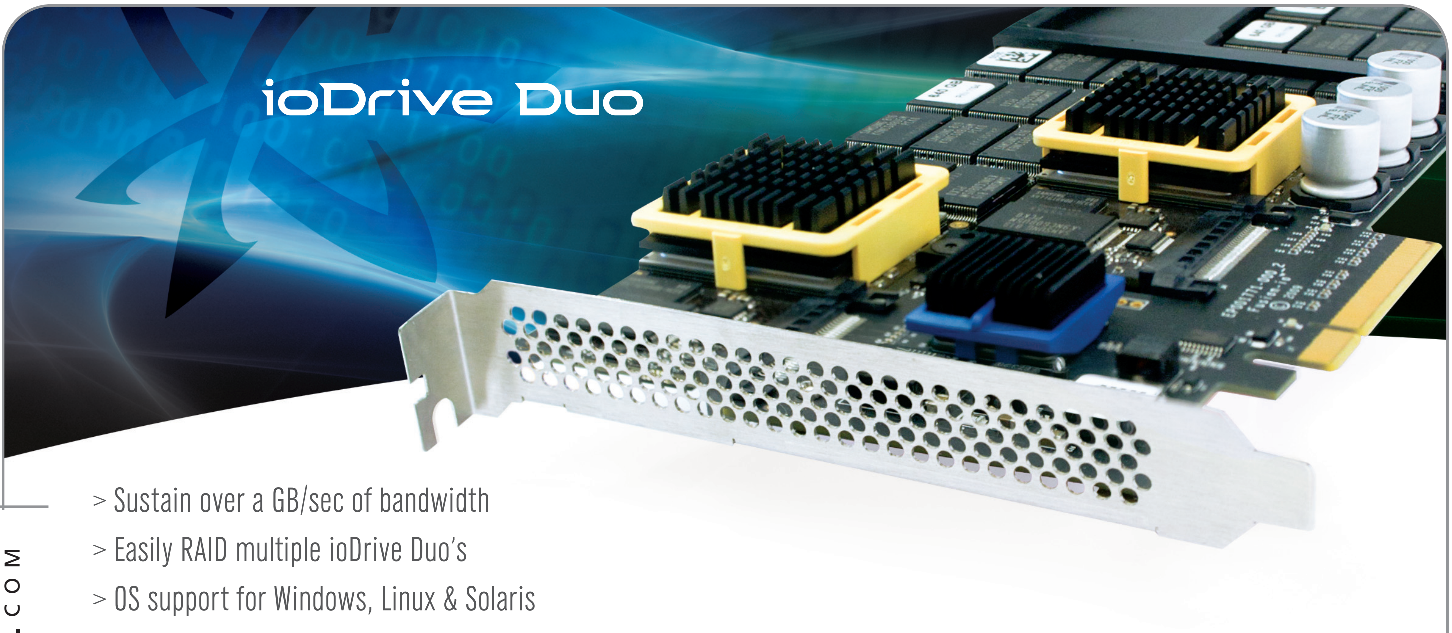
The ioDrive Duo comes in the following capacities:

- 160 Gbytes
- 320 Gbytes
- 640 Gbytes
- 1.28 TB (second half of 2009)

The ioDrive Duo will be available in April 2009. To find out more about how this and Fusion-io's other enterprise solid-state storage products can benefit your organization, please visit www.fusionio.com.

About Fusion-io

Fusion-io is a leading provider of enterprise solid-state technology and high-performance I/O solutions. The company's solid-state storage technology closes the gap between processing power and storage needs delivering breakthrough performance at a fraction of the cost of traditional disk-based storage systems. The result is a world of possibilities for performance-starved applications.



> Sustain over a GB/sec of bandwidth
> Easily RAID multiple ioDrive Duo's
> OS support for Windows, Linux & Solaris

W W W . F U S I O N I O . C O M

ioDrive Duo Capacity	160GB	320GB	640GB
NAND Type	Single Level Cell (SLC)	Single Level Cell (SLC)	Multi Level Cell (MLC)
Write Bandwidth	1.1 GB/s (32k packet size)	1.4 GB/s (32k packet size)	1.0 GB/s (32k packet size)
Read Bandwidth	1.5 GB/s (32k packet size)	1.5 GB/s (32k packet size)	1.4 GB/s (32k packet size)
IOPS*	200,832 reads (4k packet size) 132,118 writes 4k packet size)	185,022 reads (4k packet size) 167,784 writes (4k packet size)	126,601 reads (4k packet size) 180,530 writes (4k packet size)
Access Latency	50µs Read	50µs Read	80µs Read
Bus Interface	PCI-Express x8 and PCI Express 2.0 x4	PCI-Express x8 and PCI Express 2.0 x4	PCI-Express x8 and PCI Express 2.0 x4
Weight	Less than 10 ounces	Less than 10 ounces	Less than 10 ounces
Operating Systems	Microsoft Windows**, Open Solaris 10 Solaris 10, RHEL 4 & 5; SLES 9 & 10	Microsoft Windows**, Open Solaris 10 Solaris 10, RHEL 4 & 5; SLES 9 & 10	Microsoft Windows**, Open Solaris 10 Solaris 10, RHEL 4 & 5; SLES 9 & 10
Wear Leveling and Sophisticated ECC (@ 5-TB write-erase / day)	24yrs	48yrs	16yrs

** Performance achieved using multiprocessor enterprise server ** 64-Bit Windows XP, Vista, Server 2003 & 2008*

STANDARDS

Form Factor	Full height, 3/4 length PCI Express 2.0
Connectivity	PCI Express electromechanical spec 2.0
Power	PCI Express power spec 2.0

AGENCY

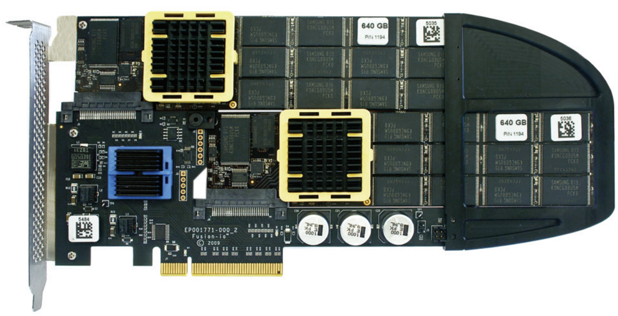
US / Canada	FCC Part 15, ICES-003, Class A
Europe	2004/108/EC EMC Directive CE Mark;
Japan	VCCI, Class A
Taiwan	BSMI, Class A
New Zealand /Australia	AS/NZS 3548 Class A
RoHS	R5 (Directive 2002/95/EC)

ENVIRONMENTAL SPECIFICATIONS

		Min	Max
Temperature (°C)*	Operational	0	55
	Non-operational	- 40	70
Air Flow (LFM)		300	
Humidity (%)	Non-condensing	5	95
Altitude (ft)	Operational		10,000
	Non-operational		30,000

** Temperature derated 1 C per 1000 ft elevation above sea level*

100% Assembled in the U.S.A.



FUSION-iO

> Less than 50 μs latency
> Easily RAID multiple ioDrives together
> Managed like simple block storage

WWW.FUSIONIO.COM

ioDrive Capacity	80GB	160GB	320GB
NAND Type	Single Level Cell (SLC)	Single Level Cell (SLC)	Multi Level Cell (MLC)
Write Bandwidth	550 MB/s (random 16K)	600 MB/s (random 16K)	500 MB/s (random 8K)
Read Bandwidth	700 MB/s (random 16K)	700 MB/s (random 16K)	700 MB/s (random 32K)
IOPS*	102,000 (random 4k reads) 91,000 (random 4k writes) 88,000 (70/30 random 4k mix)	104,400 (random 4k reads) 103,925 (random 4k writes) 95,000 (70/30 random 4k mix)	60,000 (random 4k reads) 79,000 (random 4k writes) 65,000 (70/30 random 4k mix)
Access Latency	50μs Read	50μs Read	80μs Read
Bus Interface	PCI-Express x4	PCI-Express x4	PCI-Express x4
Weight	Less than 2 ounces	Less than 2 ounces	Less than 2 ounces
Operating Systems	RHEL 4 & 5; SLES 9 & 10 Microsoft 64-Bit Windows**	RHEL 4 & 5; SLES 9 & 10 Microsoft 64-Bit Windows**	RHEL 4 & 5; SLES 9 & 10 Microsoft 64-Bit Windows**
Wear Leveling and Sophisticated ECC (@ 5-TB write-erase / day)	24yrs	48yrs	16yrs

** Performance data provided by Medusa Labs. ** 64-Bit Windows XP, Vista, Server 2003 & 2008*

STANDARDS

Form Factor	Low profile PCI Express x4 slot (spec 1.1)
Connectivity	PCI Express x4 (electromechanical spec 1.1)
Power	PCI Express x4 (power spec 1.1)

ENVIRONMENTAL SPECIFICATIONS

		Min	Max
Temperature (°C)*	Operational	0	55
	Non-operational	- 40	70
Air Flow (LFM)		300	
Humidity (%)	Non-condensing	5	95
Altitude (ft)	Operational		10,000
	Non-operational		30,000

** Temperature derated 1 C per 1000 ft elevation above sea level*

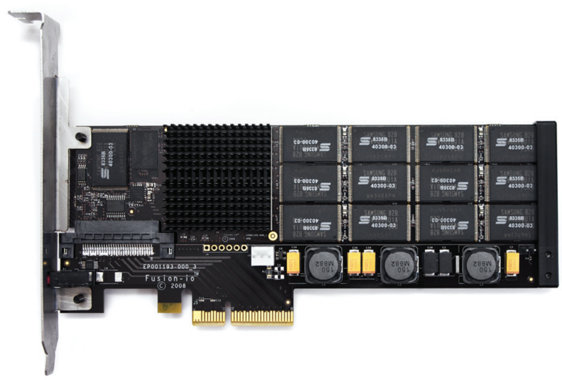
SAFETY

US / Canada	UL60950, CSA C22.2 No.60950-1-03
Europe	TUV EN60950-1:2001; 3N50825-1:

100% Assembled in the U.S.A.

AGENCY

US / Canada	FCC Part 15, ICES-003, Class A
Europe	2004/108/EC EMC Directive CE Mark;
Japan	VCCI, Class A
Taiwan	BSMI, Class A
New Zealand /Australia	AS/NZS 3548 Class A
RoHS	R5 (Directive 2002/95/EC)



FUSION-IO