WCET

Home
Research
SWEET
SWEET manual (DocBook format)
SWEET manual (PDF format)
SWEET website
Case studies
Education
Publications
Staff
Benchmarks
Downloads
Partners

MRTC
MDH
Internal

Benchmarks

The Mälardalen WCET research group maintains a large number of WCET benchmark programs, used to evaluate and compare different types of WCET analysis tools and methods. The benchmarks are collected from several different research groups and tool vendors around the world. Each benchmark is provided as a C source file (file.c). Each benchmark in the table links to a directory with the source file, a call graph (file.cg.pdf) and a scope hierarchy graph (file.sgh.pdf) as Acrobat Reader files. The scope hierarchy graph is a context sensitive graph showing calls to functions and entries to loops. Read here about scope hierarchy graphs. The graphs are produced by our WCET analysis tool SWEET.

We have also added precompiled binaries (in COFF format) for the Renesas H8300 processor. We have used gcc with these parameters. We plan to add more binaries for different targets and some informative text files.

The text file file.facit.txt describes the result of a loop analysis based on the execution of an instrumented version of the benchmark program (i.e. no overestimation).

The following table shortly presents and classifies each benchmark. You can also download a zip file containing all WCET benchmarks scource files. The benchmark programs that were selected for the WCET Challenge 2006 are marked with *.

All benchmark programs are, when run as they are, single path programs. For some of the benchmarks, we have added a text file (file.ann) which contains information of an annotation (interval of possible values) that can be made to make the program multipath. The file is in SWEET format, but the comments explains the annotation.

Do not hesitate to contact us if you want to contribute some WCET benchmarks to our selection. NOTE: we also appreciate that, if you use any of our benchmarks in a scientific publication, you refer to our benchmarks using this reference.

Legend: I = uses include files. **E** = calls external library routines. **S** = always single path program (no potential flow depenency on external variables, for a discussion, see here). **L** = contains loops. **N** = contains nested loops, **A** = uses arrays and/or matrixes. **B** = uses bit operations. **R** = contains recursion. **U** = contains unstructured code. **F** = uses floating point calculation (some of the benchmarks define floating point variables that are used for time measurements, these have not been marked. These variables can be removed). Bytes = size of source code file. **LOC** = lines of source code.

Benchmark	Description	Comments	I	E	S	L	N	A	B	R	U	F	Bytes	LOC
adpcm*	Adaptive pulse code modulation algorithm.	Completely well-structured code.				√							26852	879
bs	Binary search for the array of 15 integer elements.	Completely structured.				√		√					4248	114
bsort100	Bubblesort program.	Tests the basic loop constructs, integer comparisons, and simple array handling by sorting 100 integers				√	√	√					2779	128
cnt*	Counts non-negative numbers in a matrix.	Nested loops, well-structured code.				√	√	√					2880	267
compress*	Data compression program.	Adopted from SPEC95 for WCET-calculation. Only compression is done on a buffer (small one) containing totally random data.				√	√	√					13411	508
cover*	Program for testing many paths.	A loop containing many switch cases.			√	√							5026	240
crc*	Cyclic redundancy check computation on 40 bytes of data.	Complex loops, lots of decisions, loop bounds depend on function arguments, function that executes differently the first time it is called.			√	√		√	√				5168	128
duff*	Using "Duff's device" from the Jargon file to copy 43 byte array.	Unstructured loop with known bound, switch statement			√	√					√		2374	86
edn*	Finite Impulse Response (FIR) filter calculations.	A lot of vector multiplications and array handling.			√	√	√	√	√				10563	285
expint	Series expansion for computing an exponential integral function.	Inner loop that only runs once, structural WCET estimate gives heavy overestimate.			√	√	√						4288	157
fac	Calculates the faculty function.	Uses self-recursion.			√	√				√			332	27
fdct	Fast Discrete Cosine Transform.	A lot of calculations based on integer array elements.			√	√		√	√				8863	239
fft1	1024-point Fast Fourier Transform using the Cooly-Turkey algorithm.	A lot of calculations based on floating point array elements.			√	√	√	√				√	6244	219
fibcall	Simple iterative Fibonacci calculation, used to calculate fib(30).	Parameter-dependent function, single-nested loop			√	√							3499	72
fir	Finite impulse response filter (signal processing algorithms) over a 700 items long sample.	Inner loop with varying number of iterations, loop-iteration dependent decisions.				√	√	√					11965	276
insertsort*	Insertion sort on a reversed array of size 10.	Input-data dependent nested loop with worst-case of (n^2)/2 iterations (triangular loop).				√	√	√					3892	92
janne_complex*	Nested loop program.	The inner loops number of iterations depends on the outer loops current iteration number.			√	√	√						1564	64
jfdctint	Discrete-cosine transformation on a 8x8 pixel block.	Long calculation sequences (i.e., long basic blocks), single-nested loops.			√	√		√					16028	375
lcdnum	Read ten values, output half to LCD.	Loop with iteration-dependent flow.				√			√				1678	64
lms	LMS adaptive signal enhancement. The input signal is a sine wave with added white noise.	A lot of floating point calculations.			√	√		√				√	7720	261
ludcmp	LU decomposition algorithm.	A lot of calculations based on floating point arrays with the size of 50 elements.				√	√	√				√	5160	147
matmult*	Matrix multiplication of two 20x20 matrices.	Multiple calls to the same function, nested function calls, triple-nested loops.			√	√	√	√					3737	163

minver	Inversion of floating point matrix.	Floating value calculations in 3x3 matrix. Nested loops (3 levels).		√	√	√	√			√	5805	201
ndes*	Complex embedded code.	A lot of bit manipulation, shifts, array and matrix calculations.			√		√	√			7345	231
ns*	Search in a multi-dimensional array.	Return from the middle of a loop nest, deep loop nesting (4 levels).			√	√	√				10436	535
nsichneu*	Simulate an extended Petri Net.	Automatically generated code containing large amounts of if-statements (more than 250).			√						118351	4253
prime	Calculates whether numbers are prime.	Uses integer division and modulo function.		√	√						797	47
qsort-exam	Non-recursive version of quick sort algorithm.	The program sorts 20 floating point numbers in an array. Loop nesting of 3 levels.			√	√	√			√	4535	121
qurt	Root computation of quadratic equations.	The real and imaginary parts of the solution are stored in arrays.		√	√		√			√	4898	166
recursion*	A simple example of recursive code.	Both self-recursion and mutual recursion are used.		√					√		620	41
select	A function to select the Nth largest number in a floating point array.	A lot of floating value array calculations, loop nesting (3 levels).			√	√	√			√	4494	114
sqrt	Square root function implemented by Taylor series.	Simple numerical calculation.		√	√					√	3567	77
st	Statistics program.	This program computes for two arrays of numbers the sum, the mean, the variance, and standard deviation, and the correlation coefficient between the two arrays.	√		√		√			√	3857	177
statemate*	Automatically generated code.	Generated by the STAtechart Real-time-Code generator STARC.			√						52618	1276
ud	Calculation of matrixes.	Loop nesting of 3 levels.		√	√	√					6 K	163

Responsible for the information on this page: jan.gustafsson@mdh.se. Latest change: 2012-10-03