WCET Analysis Lab: Assignment 1

Markus Klein
Johannes Kasberger

SS 2012

Problem 1
Recommended (3): As a warm-up exercise, follow the instruction in Timing Analysis Lab: First Steps
Q: How long does it take to execute simple once, according to measurements, and according to the static analysis?

Answer

- Measurements: 763 incl. function call - 70 overhead = 693 cycles
- Static analysis: 705 cycles

The difference is 12 cycles so the measurement is not far away from the static analysis.

Problem 2
Recommended (3): Also extract the instruction trace as outlined in Timing Analysis Lab: First Steps. Then compare the number of cycles needed in one iteration of the loop, with the number of cycles aiT calculated.
Q: Do they coincide? What is the total number of cycles needed to execute simple according to the instruction trace buffer?

Answer

Problem 3
Mandatory (4): First, create a project containing the files contained in the insertion sort folder of the task specification. Now complete the function main.c:run(), executing insertion sort a few times, with array size 32 and different input data. Measure the minimum and maximum time needed to execute the sort function.
Q: What were the results of the measurement? How many test sets would do you need to cover all possible execution path?

Answer

Problem 4
Mandatory (5): Add loop bounds and additional flow facts for insertion sort.c:insertion sort(), using the symbolic name @size for the size of the array to be sorted. Next, analyze the WCET of insertion sort, assuming an array size of 32. Keep the array size as a symbolic name (user register @size). Finally, write a test function which calls insertion sort more than once, with different array sizes (e.g., 16,32 and 64). Also repeat the static analysis with different array sizes.
Q: How many cycles do you need to execute insertion sort according to the static analysis?

Answer

Q: What results do you get for an array size of 8,16 or 64, using measurements and static analysis?

Answer

Q: In addition to the size of the array, what other aspects of the input data might influence the WCET?

Answer

Problem 5
Recommended (8 Points): Assume that your goal was to find out the WCET of task.c:task(). Before analyzing the execution time, you should answer a few basic questions about the input data for the monitoring task, and analyze the control flow on the source code level.
Q: What is the set of input data which might influence the execution time of the task at the software side? Is it tractable to enumerate every possible input? Which loops need to be bounded? Add all loop bounds and flow facts you can find to the file task.c (as source code annotations).

Answer

Problem 6
Recommended (8 Points): Analyze the fft() function called in task.c. Try to find loop bounds for the Fast Fourier Transform implementation (fixedpoint.c:fp radix2fft with-scaling) first. If you have difficulties finding them, add a debug statement and run the transform with different input data sizes. Add flow constraints relating the execution frequency of the inner loops with the functions execution frequency. Finally, try to analyze the execution time using aiT. There is already a timing measurement for the fft in the executable, so it is easy to compare the number of cycles estimated to execute the function.
Q: Compare the worst-case number of iterations for the inner loop with and without using these flow constraints. Finally, think about the complexity of calculating loop bounds for

FFT.

Answer

Q: Does the FFT loop bound depend on the input data?

Answer

Problem 7
Optional Challenge (5 Bonus Points): Try to analyze the WCET of task.c:task() using aiT. If you attempt to solve this challenge, use the control flow graph and disassembling capabilities of aiT, and be sure to understand the source code you are analyzing.
Problem 8
Mandatory (4): Answer the following questions
Q: How much time did you spend writing annotations and analyzing the code? Was it less or more than you expected? How much time did you spend on this first assignment?

Answer

Q: What is the ratio between observed and actual execution time? Discuss the causes of the overestimation.

Answer

Q: As you learned, sometimes it is necessary to annotate the assembler code. Why? What problems can you see because of this?

Answer