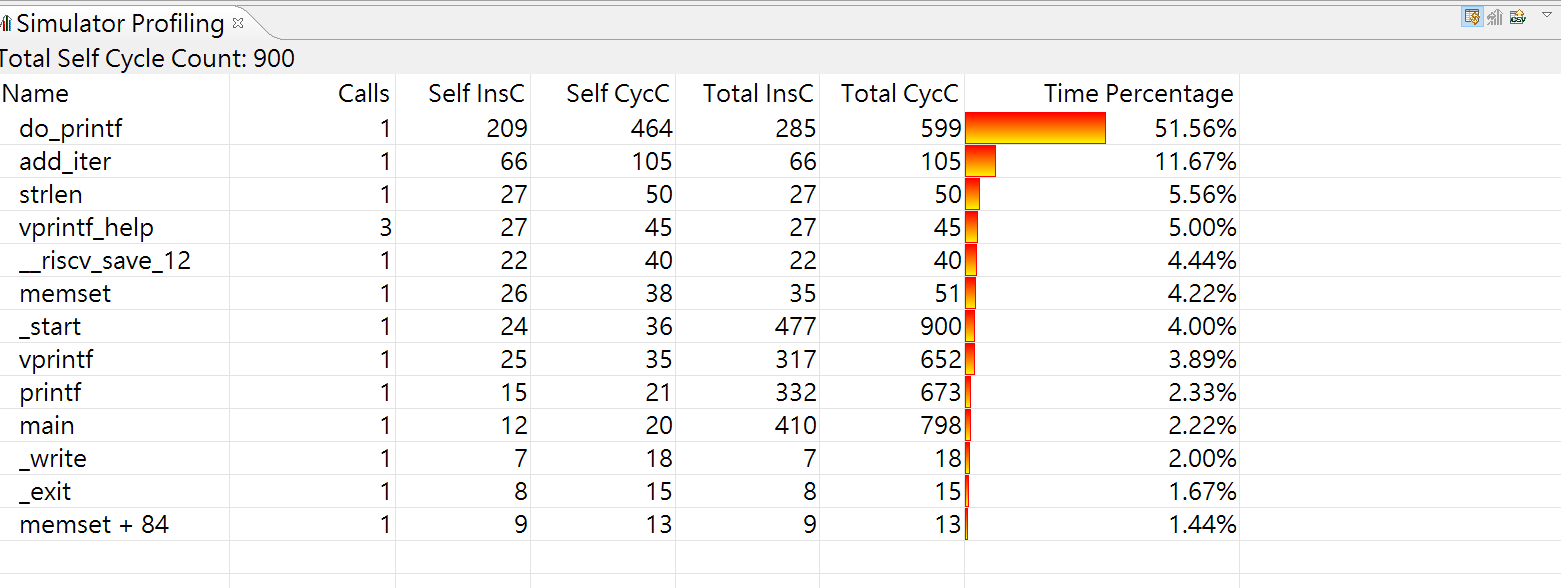
**109062320 朱季葳 , discuss with 109062233, 109062306, 109062123,109062323**

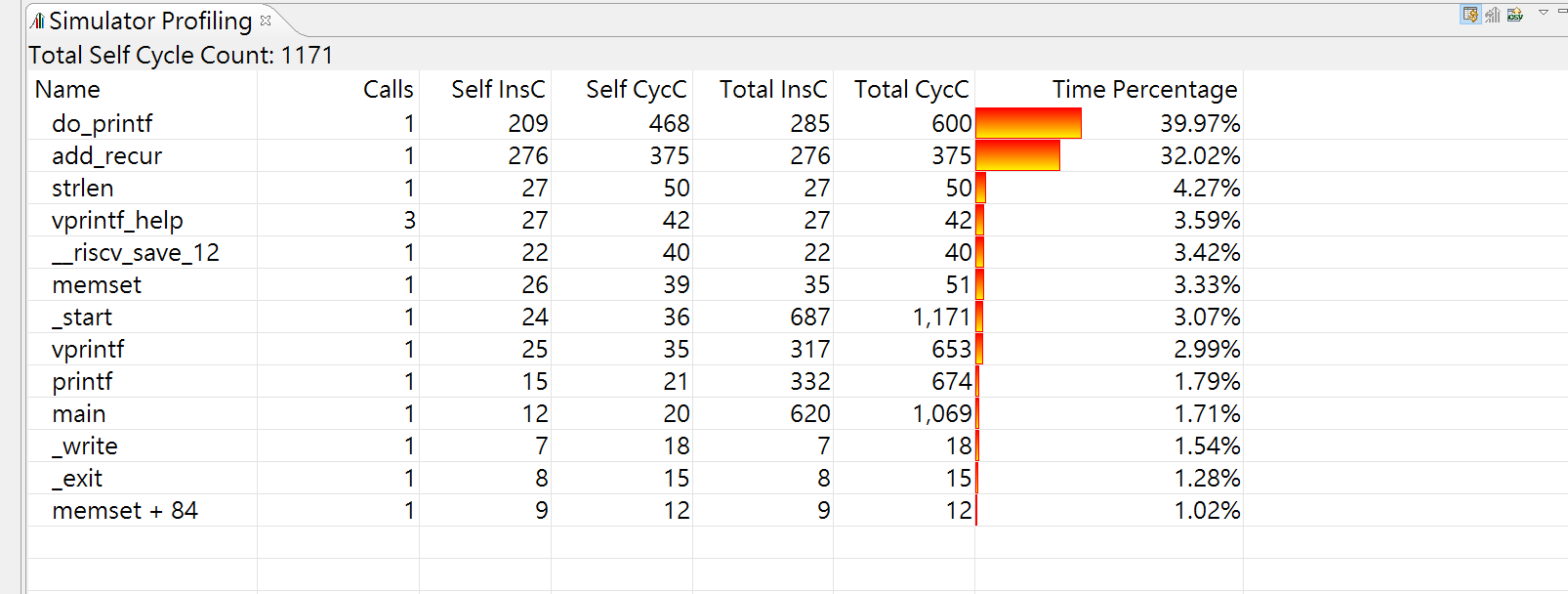
1.

(a) (6 points) Record the results for the two programs using CycC and InsC for “add\_recur” function of “ADD\_recursive” and for “add\_iter” function of “ADD\_iterative”. Based on your understanding of the characteristics of the two programs, compare the differences in their profiles.



**result of ADD\_iterative**

**add\_iter :InsC 66 CycC 105**



**result of ADD\_recursive**

**add\_recur :InsC 276 CycC 375**

**By the above graph,we can see that the instruction count and clock cycle count of recursion are bigger than the ones of iteration**

**and the CPI of recursion is 375/276 = 1.358, the CPI of iteration is 105/66 = 1.59**

**As a result, we can see that the CPI of iteration is bigger**

**Furthermore, as we know about recursion, it calls it self until the terminal condition, and it is far more time-consuming than iteration in which just run a loop.**

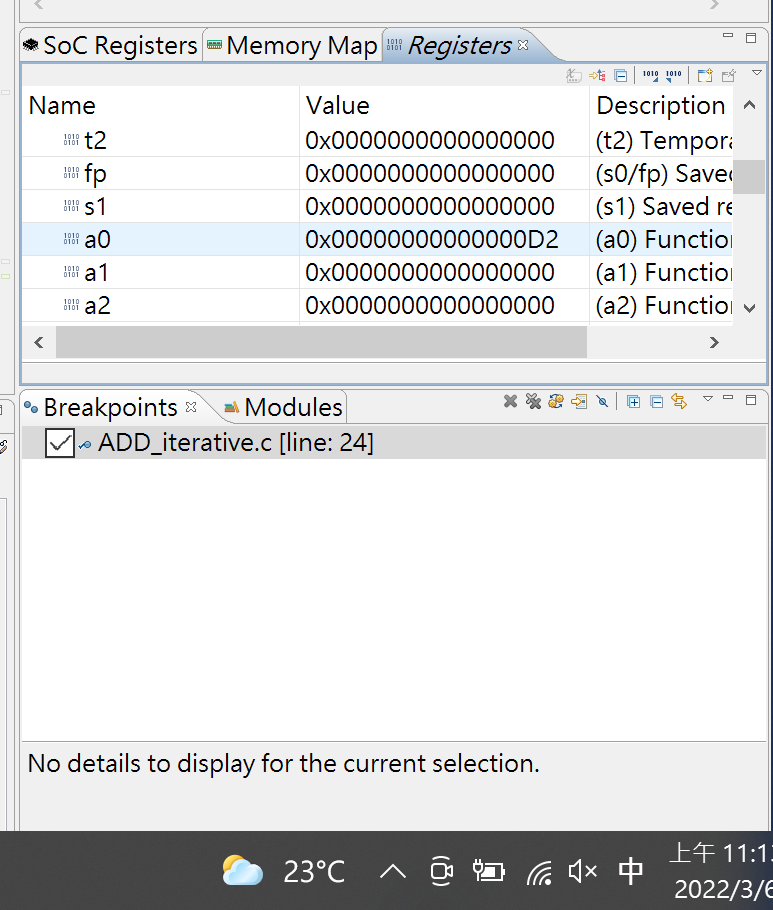
(b)(6 points) RISC-V has 32 general purpose registers, whereas X86 (a CISC architecture) only has 8. Why does RISC have more registers than CISC?

**The primary goal of CISC architecture is to complete a task in as few lines of assembly code as possible while RISC architecture only use simple instructions that can be executed within 1 clock cycle.**

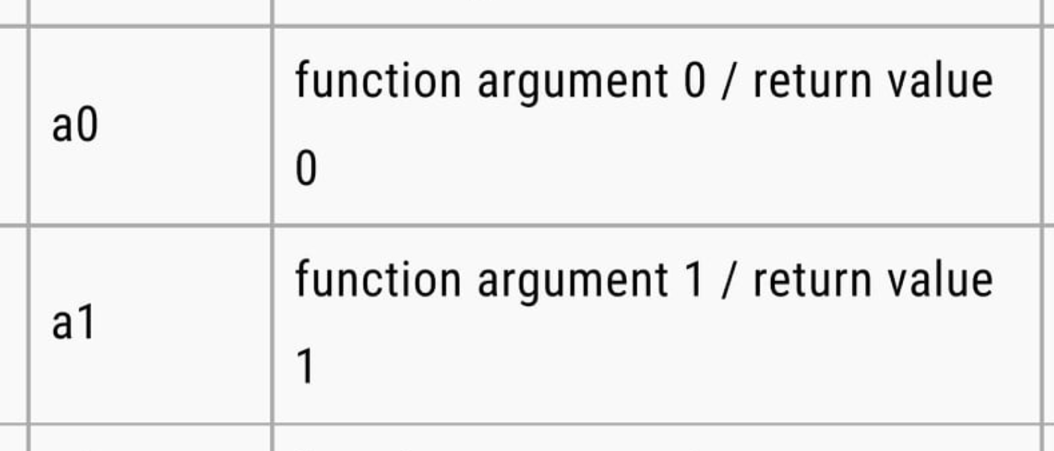
**As a result, RISC need more registers to store the instructions.**

**I take reference on** [**https://cs.stanford.edu/people/eroberts/courses/soco/projects/risc/risccisc/**](https://cs.stanford.edu/people/eroberts/courses/soco/projects/risc/risccisc/)

When executing “ADD\_iterative”, which register stores the return value for add\_iter function? (You can select debug icon to view the contents of all registers. Note that you should set breakpoints at the right spots.)



**By the above image, after set breakpoint in the line after calling function,we can see the register “a0” store the return value for add\_iter function**



**Besides, by the description of a0 and a1, we can see that a1 also store the return value**

(c) (6 points) What are the average CPI for “add\_recur” function in ADD\_recursive.c and “add\_iter” function in ADD\_iterative.c, respectively?

**CPI of recursion is 375/276 = 1.36, the CPI of iteration is 105/66 = 1.59**

(d) (6 points) What are the CPU execution time for “add\_recur” function in ADD\_recursive.c and “add\_iter” function in ADD\_iterative.c, respectively, on a processor with a clock rate of 1GHz?

**execution time of recursion is 375/(1\*10^9) = 0.000000375 = 0.375ms ~= 0.38ms**

**execution time of iteration is 105/(1\*10^9) = 0.000000105 = 0.105ms ~= 0.11ms**

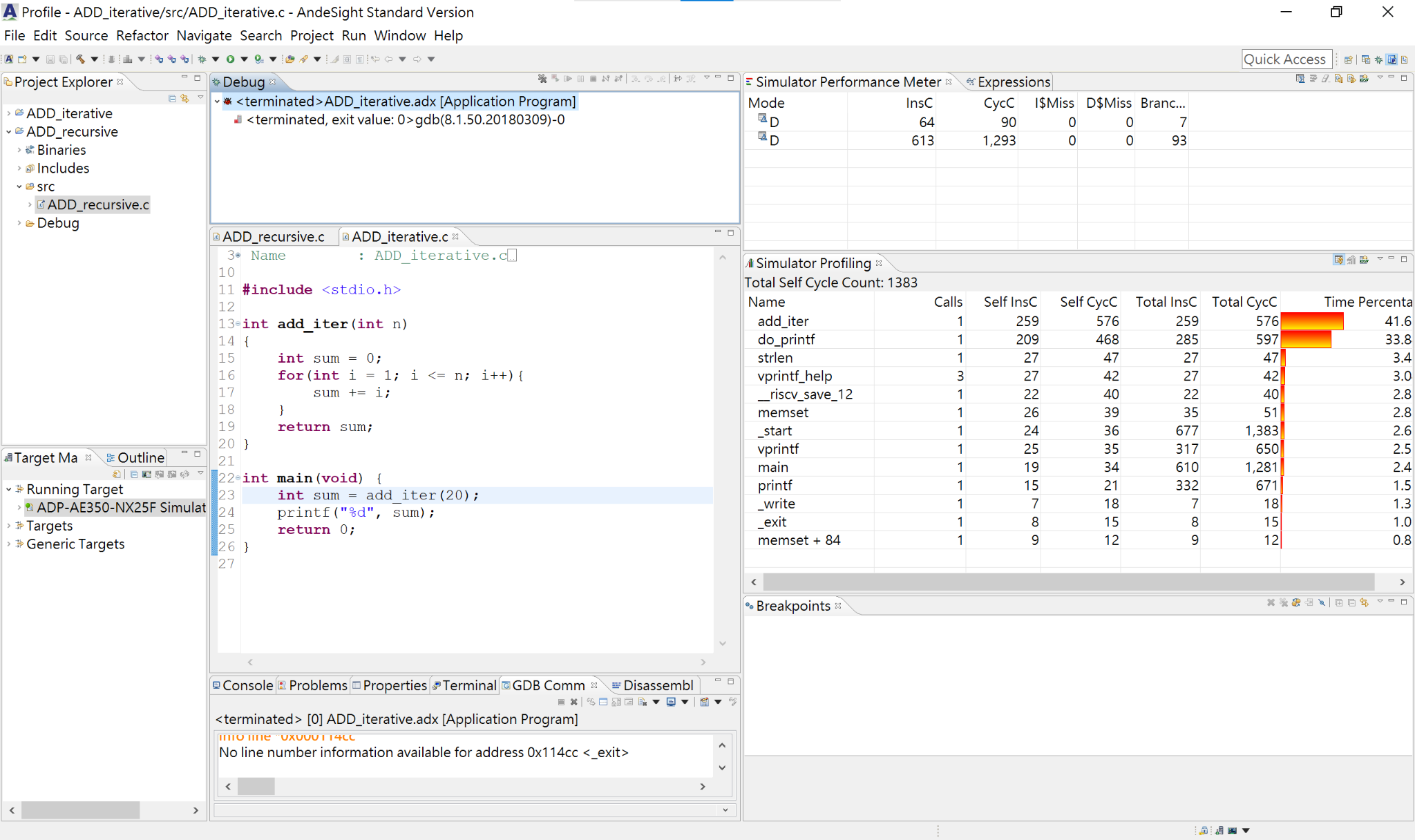
(e) (6 points) Assume we execute ADD\_iterative.c in (d) but with a 4-core multiprocessor instead and also employ parallelization techniques on add\_iter function to equally allocate computation to each core. But, the parallelization increases 50 communication cycles on the multiprocessor. What is the program execution time now? (Hint: Only add\_iter can be parallelized.)

**total cycle = 900**

**clock cycle count = 900 – 105 + 50 + 105/4 = 872**

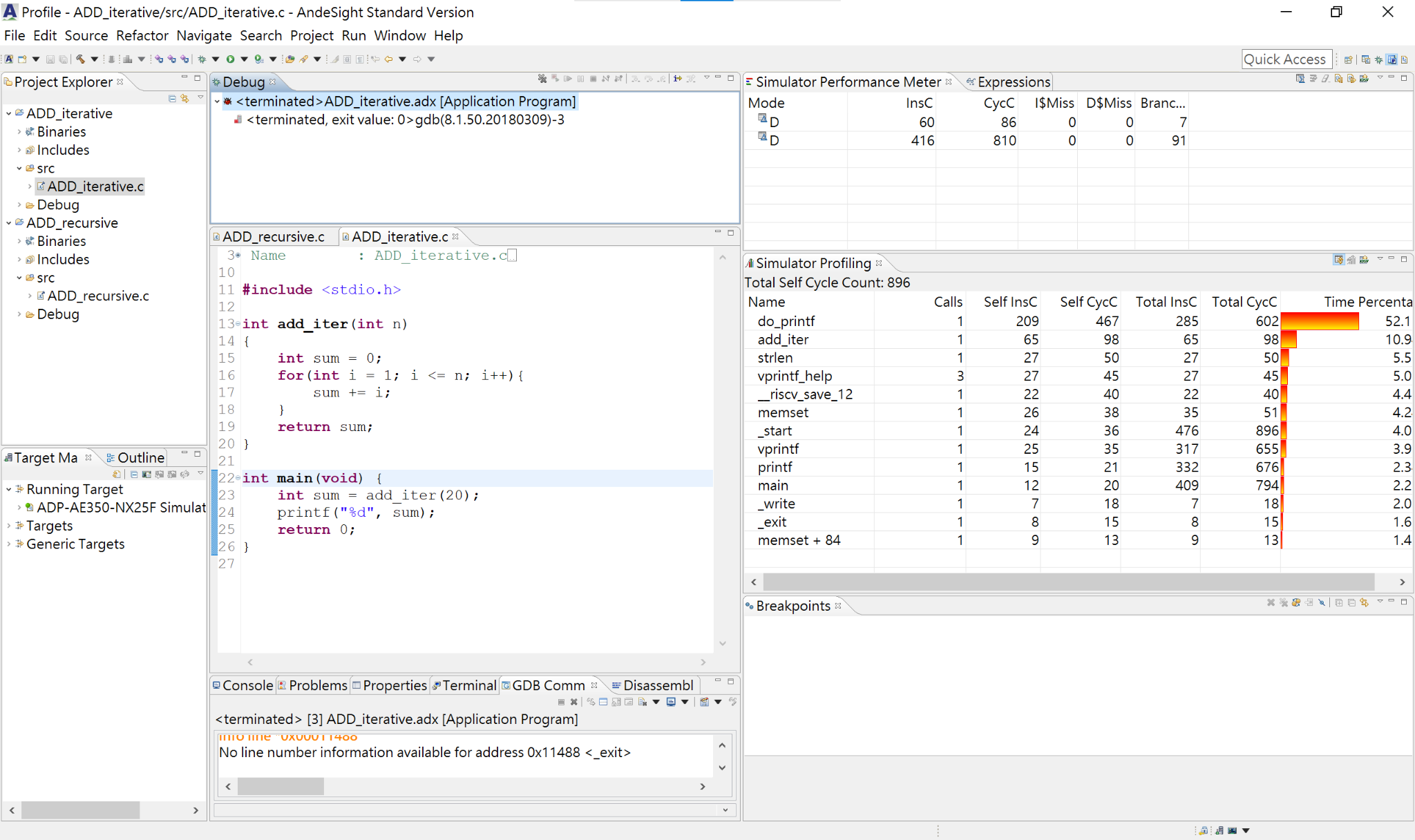
**execution time = 872 / 10^9 = 0.872 ms ~= 0.87ms**

(f) (20 points) Compiler will affect program performance. Compile ADD\_recursive.c and ADD\_iterative.c with two optimization levels, -O0 and -O1, respectively. Compare the performance of “add\_recur” and “add\_iter” function in ADD\_recursive.c and ADD\_iterative.c for different optimization levels (-O0 and -O1). You should report CycC, InsC and CPI and explain the differences in their profiles. (To change optimization level, see Fig. 1.)

****

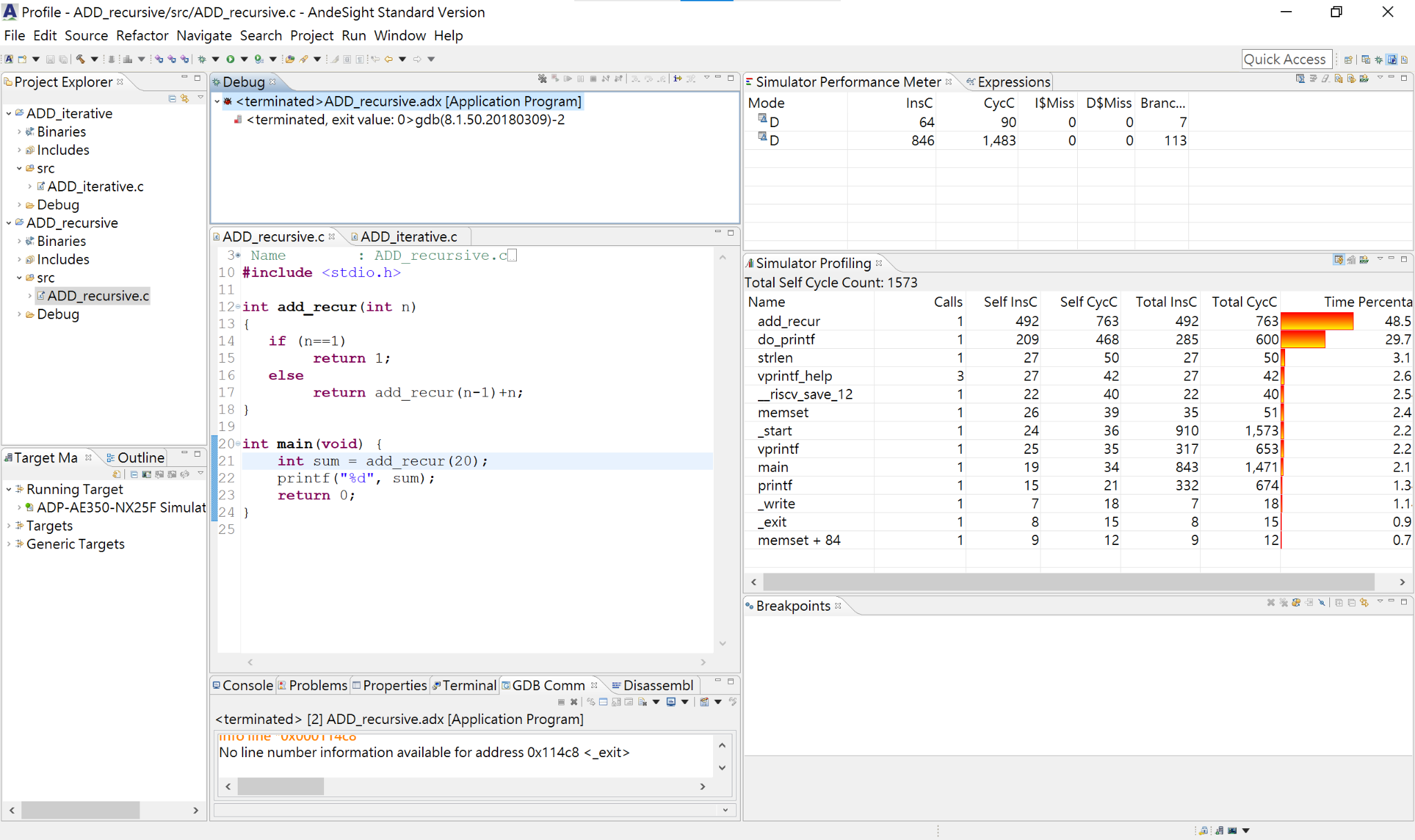
**result of iteration with -O0**

**CycC: 576, InsC: 259, CPI: 576/259 = 2.2239 ~= 2.22**

****

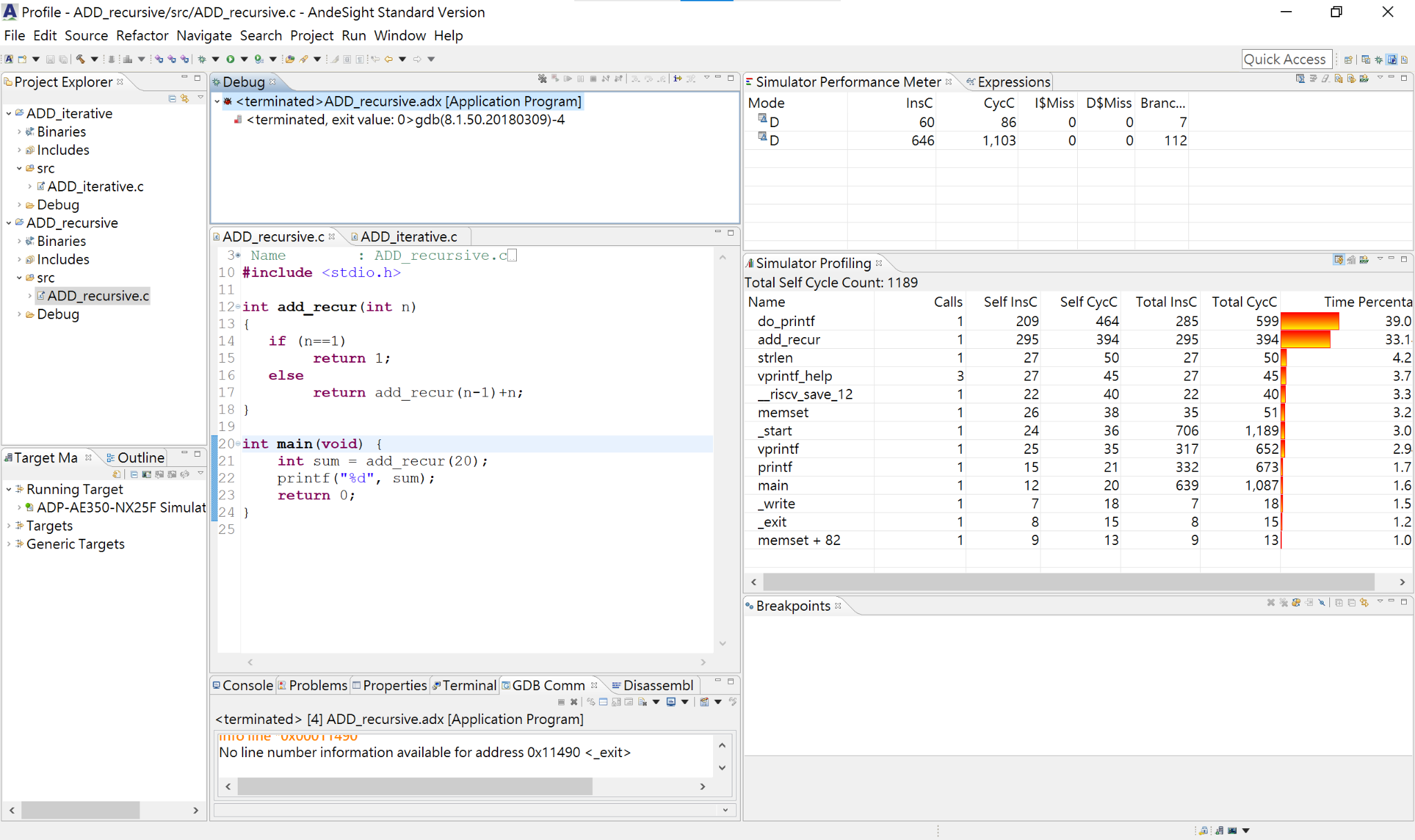
**result of iteration with -O1**

**CycC: 98, InsC: 65, CPI: 98/65 = 1.5076 ~= 1.51**

****

**result of recursion with -O0**

**CycC: 763, InsC: 492, CPI: 763/492 = 1.5508 ~= 1.55**

****

**result of recursion with -O1**

**CycC: 394, InsC: 295, CPI: 394/295 = 1.3355 ~= 1.34**

**By the above images, we can see that O1 has less CycC, InsC and CPI than O0.**

**I take reference on the official website:** [**https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html**](https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html)

**O0 compiler is the original one which doesn’t have any optimization,while O1 compiler optimize compilations that take more time and more memory for a large function.**

**Besides, O1 compiler will try to reduce code size and execution time without performing the optimizations that take lots of compilation time.**

**As for the original setting --> -Og**

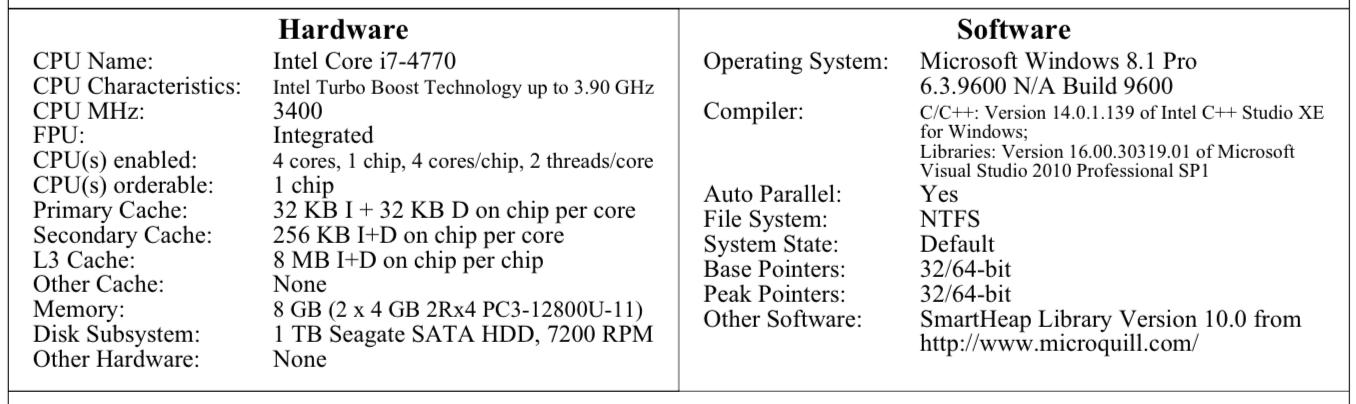
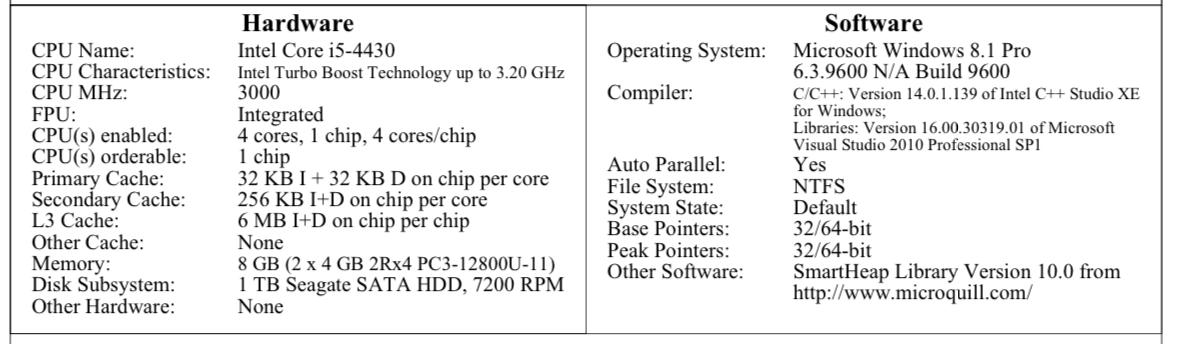
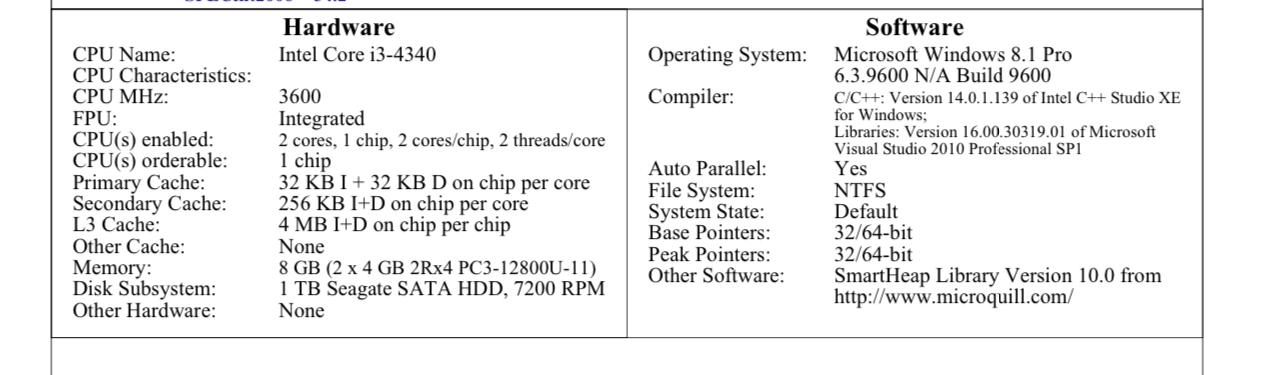
**Optimize debugging experience.  -Og  should be the optimization level of choice for the standard edit-compile-debug cycle, offering a reasonable level of optimization while maintaining fast compilation and a good debugging experience.**

**It is a better choice than -O0 for producing debuggable code because some compiler passes that collect debug information are disabled at -O0**

2. (20 points) Go to SPEC Benchmarks website and search for the benchmarks of the following systems: • Intel DH87MC Motherboard (Intel Core i3-4340) • Intel DH87MC Motherboard (Intel Core i5-4430) • Intel DH87MC Motherboard (Intel Core i7-4770)

We want to compare the performance of the three computer systems using different methods to normalize the performance.

(a) (5 points) Compare the hardware and software specifications of these three motherboards and identify their differences.



**Firstly, we compare the differences in software.We can see that there is no difference in software among these three motherboards.**

**As for the differences in hardware,there are four differences:**

**1.Turbo-Boost, i3 doesn’t has turbo-boost enabled,while i5 and i7 have;besides,i7’s speed is faster than i5.**

**2.cores, i3 has 4 cores,i5 has at least 4 cores while i7 has at least 6 cores**

**3.cache memory, the storage of cache memory in i7 > i5 > i3**

**4.the running frequency, the running frequency in i3 > i7 > i5**

**(b)** (5 points) Consider the three benchmarks: 464.h264ref, 471.omnetpp, and 473.astar in the results table.

Please use the first column of “seconds” (check Fig. 4) to calculate the relative performance of the three computer systems based on the three benchmarks.

Fill out the following table, which uses each of the three computers as the reference for comparison.

Summarize the performance results with the arithmetic mean of the performance ratios of the three benchmark programs. Please show the calculation procedure.

464.h264ref

|  |  |  |  |
| --- | --- | --- | --- |
|  | performance ratio |  |  |
| reference | intel core 13-4340 | intel core i5-4430 | intel core i7-4770 |
| intel core i3-4340 | 1 | 323/373=0.87 | 323/323=1 |
| intel core i5-4430 | 373/323=1.16 | 1 | 373/323=1.16 |
| intel core i7-4770 | 323/323=1 | 323/373=0.87 | 1 |

471.omnetpp

|  |  |  |  |
| --- | --- | --- | --- |
|  | performance ratio |  |  |
| reference | intel core 13-4340 | intel core i5-4430 | intel core i7-4770 |
| intel core i3-4340 | 1 | 257/251=1.02 | 257/223=1.15 |
| intel core i5-4430 | 251/257=0.98 | 1 | 251/223=1.13 |
| intel core i7-4770 | 223/257=0.87 | 223/251=0.89 | 1 |

473.astar

|  |  |  |  |
| --- | --- | --- | --- |
|  | performance ratio |  |  |
| reference | intel core 13-4340 | intel core i5-4430 | intel core i7-4770 |
| intel core i3-4340 | 1 | 233/247=0.94 | 233/200=1.17 |
| intel core i5-4430 | 247/233=1.06 | 1 | 247/200=1.24 |
| intel core i7-4770 | 200/233=0.86 | 200/247=0.81 | 1 |

Arithmetic mean

|  |  |  |  |
| --- | --- | --- | --- |
|  | performance ratio |  |  |
| reference | intel core 13-4340 | intel core i5-4430 | intel core i7-4770 |
| intel core i3-4340 | 1 | (1.02+0.87+0.94)/3  =0.94 | (1.15+1+1.17)/3  =1.11 |
| intel core i5-4430 | (0.98+1.16+1.06)/3  =1.07 | 1 | (1.13+1.16+1.24)/3  =1.18 |
| intel core i7-4770 | (0.87+1+0.86)/3  =0.91 | (0.89+0.87+0.81)/3  =0.86 | 1 |

(c) (5 points) Repeat (b) with the geometric mean of the performance ratios of the three benchmark programs.

|  |  |  |  |
| --- | --- | --- | --- |
|  | performance ratio |  |  |
| reference | intel core 13-4340 | intel core i5-4430 | intel core i7-4770 |
| intel core i3-4340 | 1 | (1.02\*0.87\*0.94)^(1/3)  =0.94 | (1.15\*1\*1.17) ^(1/3)  =1.10 |
| intel core i5-4430 | (0.98\*1.16\*1.06)^(1/3)  =1.06 | 1 | (1.13\*1.16\*1.24) ^(1/3)  =1.18 |
| intel core i7-4770 | (0.87\*1\*0.86) ^(1/3)  =0.91 | (0.89\*0.87\*0.81) ^(1/3)  =0.86 | 1 |

(d) (5 points) Which observations can you make from (b) and (c)? How are these results compared with SPECint\_base2006 ratio?

**We can see that there is little difference between geometric mean and arithmetic mean;what’s more,it shows that the performance of i7 > i3 > i5.**

**But the result of arithmetic mean looks more changable than geometrc mean.**

**After comparing with the result of SPECint\_base2006,we can also see that the performance of**

**i7 > i3 > i5 （The SPECint\_base2006 is i3 : 52.3 i5 : 48.3 i7 : 59.6 ）**

3. (10 points) Assume a program requires the execution of 90 × 106 FP instructions, 110 × 106 INT instructions, 100 × 106 L/S instructions, and 25 × 106 branch instructions. The CPI for each type of instruction is 2, 1, 5, and 2, respectively. Assume that the processor has a 4GHz clock rate.

1. (a)  (5 points) By how much must we improve the CPI of FP instructions if we want the program to run two times faster? Please show the calculation procedure.

**clock cycles = (90 x 10^6 x 2) + (110 x 10^6 x 1) + (100 x 10^6 x 5) + (25 x 10^6 x 2)**

**= (180+110+500+50)x10^6 = 840 x 10^6**

**run two times faster-->clock cycles/2 (clock rate is fixed) = 420 x 10^6**

**new CPI of FP instruction = (420 – 110 – 500 – 50) x 10^6 / (90 x 10^6)**

**= -240/ 90 --->it is impossible!**

1. (b)  (5 points) By how much is the execution time of the program improved if the CPI of INT and FP instructions is reduced by 31% and the CPI of L/S and Branch is reduced by 77%? Please show the calculation procedure.

**clock cycles = {0.69 x [(90 x 10^6 x 2) + (110 x 10^6 x 1)]}**

**+ {0.23 x [(100 x 10^6 x 5) + (25 x 10^6 x 2)]}**

**= (290x0.69 + 550x0.23)x10^6 = 326.6 x 10^6**

**because clock rate is fixed: 326.6/840 = 0.388 ~= 39% improved**

**so the improvement is 61%**

1. (10 points) Processor P1 has a clock rate of 5GHz and a voltage of 1.25V. Assume that, on average, it consumes 60W of static power and 90W of dynamic power. Assume that the dynamic power and static power are respectively calculated by the following equations:

dynamic power = 1/2 × capacitive load × volatge2 × clock frequency

static power = voltage × leakage current

* + 1. (2 points) Find the average capacitive load.

**capative load = 2 x dynamic power / (voltage^2 x clock frequency)**

**P1:capative load = 2 x 90 / ( 1.25^2 x 5\*10^9)**

**= 2.30 x 10^(-8)**

* + 1. (2 points) Find the percentage of the total dissipated power comprised by dynamic power and the ratio of static power to dynamic power.

**percentage of total dissipated power comprised by dynamic power 90 / (90 + 60) = 0.6**

**ratio of static power to dynamic power = 60/90 = 0.67**

* + 1. (6 points) If the total dissipated power is to be reduced by 35%, how much should the voltage be reduced to maintain the same leakage current?

**abbriviate Static power,dynamic power,voltage,capative load,**

**clock frequency,leakage current to S,D,V,C,F,L**

**(Snew + Dnew) / (Sold + Dold) = 0.65, Vnew = (2 x Dnew / (C x F))^(1/2), Dnew = 0.65 x (Sold + Dold) – Snew**

**Snew = Vnew x L ,Sold = Vold x L -->Snew/Vnew = Sold/Vold**

**L = 60 / 1.25 = 48**

**Dnew = 0.65 x (60+90) – Vnew x 48 = 97.5 – Vnew x 48**

**Vnew = [(2 x 97.5 – Vnew x 96)/(2.3 x 10^(-8) x 5 x 10^(9))]^(1/2) = 0.95**

**Vold = 60 / 48 = 1.25**

**Vnew/Vold = 0.95/1.25 = 0.76, the voltage reduced by 24%**

1. (10 points) Assume that a 25 cm diameter wafer has a cost of 15, contains 120 dies, and has 0.02 defects/cm2.
2. (3 points) Find the yield for this wafer using the equation on page 28 of the textbook

(or page 28 of the lecture notes of Chapter 1).

**Die area = pi x 12.5^2 / 120 = 4.09**

**Yield = 1/((1+Defects per area x Die area/2))^2**

**= 1/((1 + 0.02 x 4.09/2))^2**

**= 0.92**

1. (3 points) Find the cost per die for this wafer.

**15/(120 x 0.92) = 0.14**

1. (4 points) If the number of dies per wafer is increased by 20% and the defects per area

unit increases by 35%, find the die area and yield.

**number of dies per wafer = 120\*1.2 = 144**

**die area = pi x 12.5^2 / 144 = 3.41**

**Yield = 1/((1 + 0.02 x 1.35 x 3.41/2))^2 = 0.91**