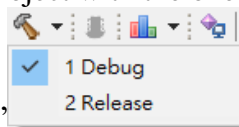










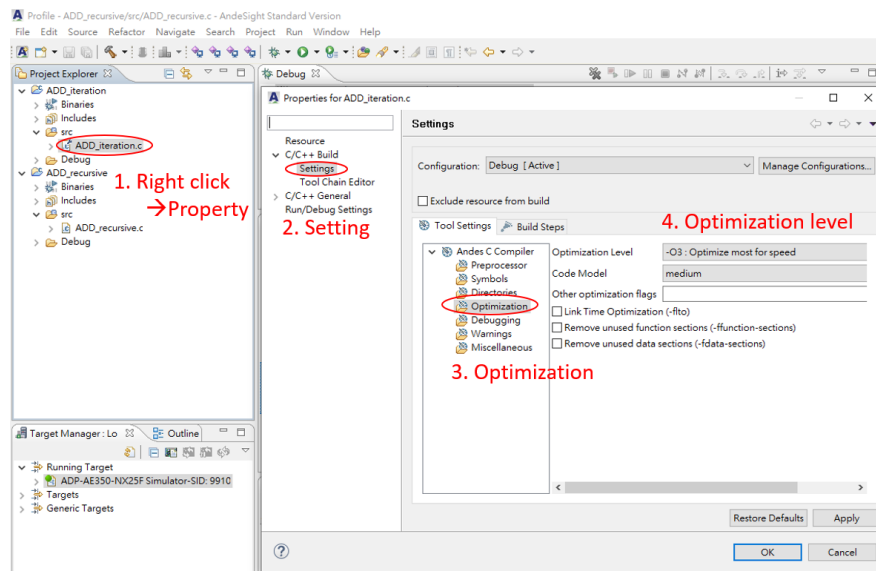
**Department of Computer Science**  
**National Tsing Hua University**  
**EECS403000 Computer Architecture**  
Spring, 2023, Homework 1  
Due date: ???

1. (50 points) Install and use AndeSight™ for RISC-V program development.
  - (1) See AndeSight\_STD\_v3.2.0\_Installation\_Guide.pdf to install AndeSight™.
  - (2) Create a new Andes C project:
    - (i) File → New → Project, select “Andes C project”.
    - (ii) Project name: “Factorial\_recursive”.
    - (iii) Chip profile name: AE350 → “ADP-AE350-NX25F”.
    - (iv) Project type: Andes Executable → “Hello World ANSI C Project”.
    - (v) Tool chains: nds64le-elf-mculib-v5d.
    - (vi) All the others remain the same.
  - (3) Replace Factorial\_recursive.c in the project with the one on eLearn.



- (4) Press the button “Build”  “Debug”  for project “Factorial\_recursive” in the toolbar.
  - (5) Do the same for project “Factorial\_iterative”.
  - (6) Press the button “Debug”  in the toolbar.
  - (7) Press the button “Resume”  in the debug window.
- Use default optimization setting -Og (Check Fig. 1 below) and answer the following questions.

- (a) (6 points) *Effects of application program (programming style) on performance.* Press the button “Profile”  in the toolbar and profile as “Application Program”. Press the button “Resume”  in the debug window and record the results for the two programs: CycC and InsC for the **iter()** function of **Factorial\_iterative.c** and the **recur()** function of **Factorial\_recursive.c**. Note that **Factorial\_iterative.c** iterates through the factorial computation with a for-loop, while **Factorial\_recursive.c** with function calls. Based on this characteristic, compare and explain the differences in their profiles.
- (b) (6 points) *Effects of application program on performance.* What are the average CPI and CPU execution time for the **recur()** function in **Factorial\_recursive.c** and **iter()** function in **Factorial\_iterative.c**, respectively, if they are executed on a processor with a clock rate of 2GHz?
- (c) (6 points) *Effects of compiler on performance.* From (a), we observe that function calls may be inefficient to execute. But, they are very common in programming. To “make common cases fast”, let us see how compilers can do. One thing that compilers can help is to allocate application program variables, which are memory locations, to processor registers. This is because registers are faster than memory. When executing **Factorial\_recursive.c**, which register stores the return value of the **recur()** function? Which register stores the function parameter **n**? (Press the button “Debug”  in the toolbar. In the Debug view, you can check the contents of all registers. Note that you should set breakpoints at the right position. Press the button “Resume”  in the debug window and you can go to the next breakpoint.)
- (d) (10 points) *Effects of compiler on performance.* Compile **Factorial\_iterative.c** and **Factorial\_recursive.c** with two optimization levels, -O0 and -O1, respectively. Compare the performance of **iter()** and **recur()** function in **Factorial\_iterative.c** and **Factorial\_recursive.c** for different optimization levels (-O0 and -O1). You should report CycC, InsC and CPI and explain the differences in their profiles. To change the optimization level, see the figure below.



(e) (12 points) *Effects of instruction set architecture (ISA) on performance.*

- What is the difference between RISC and CISC?
- Use the keyword “more”, “less”, and “same” to fill in the blanks in the table below to show **qualitatively** how ISA design influences the performance.

	IC	CPI	Cycle time
RISC			
CISC			

- Suppose instructions handling function calls take 40% of the instructions in the **recur()** function in **Factorial\_recursive.c** but require 30% of the execution cycles. Now, suppose you develop a new instruction that can reduce the function-handling instructions in the **recur()** function by 35%, but it also increases the execution cycles of those instructions by 25%. Will you introduce this new instruction into your ISA?

(f) (10 points) *Effects of great ideas on computer performance.* Two of the great ideas in enhancing computer performance are parallelism and pipelining. Consider **Factorial\_iterative.c** and neglect all overheads due to parallelization and data transmission.

- Pipelined execution.** One way to perform pipelined execution is to use one core to increment the index **i** and then pass the index to the second core, where the operations in **sum \*= i** are executed. Examine the profile of **iter()** in **Factorial\_iterative.c** and explain whether it is possible to get a speedup of 2 for the pipelined execution of this function.  
(Hint: The cycles when executing one for-loop and the multiplication instruction within can be observed by using breakpoint.)
- Parallel execution.** Suppose we want to improve the performance of the computer with a 4-core multiprocessor. One way to parallelize the **iter()** function is to divide the computations into 3 stages. Denote the 4 cores as A, B, C, and D, respectively:
  - Assign the iterations to the 4 cores to execute as evenly as possible.
  - Core A sends its multiplication result to B and C to D. Then, B and D do multiplication again.
  - Core B sends its multiplication result to D, where the final result is calculated.
 Based on **Factorial\_iterative.c**, what is the percentage reduction of the cycle count in executing the **iter()** function using parallel execution? Can you get a speedup of 4?  
(Hint: Since each iteration executes one multiplication, you can assume the cycle count of executing one iteration is the same as that of executing one multiplication.)

2. (25 points) We want to compare the performance and battery lives of the following two mobile systems.

Product	Samsung Galaxy S23 Plus	Apple iPhone 14 Pro Max	
SoC	Snapdragon 8 Gen 2	Apple A16 Bionic	
Core*	1+2+2+3	2+4	
Web browser (Wi-Fi)	13:54 hr	15:02 hr	
Video playback	15:56 hr	21:10 hr	
Standby	118 hr	156 hr	
AnTuTu 9	1273667	939161	
Geekbench 5**	1480 / 4938	1867 / 5371	
Image compression	251.3 Mpixels/s	243.5 Mpixels/s	
Face detection	46.2 images/s	46.4 images/s	
Speech recognition	122.9 words/s	141.8 words/s	
Note			
*Heterogeneous CPU architecture, the former core the peak frequency is higher.			
**Results for the single-core / multi-core Geekbench 5 test, respectively.			
Reference:			
Performance	<a href="https://nanoreview.net/en/soc-list/rating">https://nanoreview.net/en/soc-list/rating</a>		
Time	<a href="https://nanoreview.net/en/phone-list/endurance-rating">https://nanoreview.net/en/phone-list/endurance-rating</a>		

- (a) (5 points) Follow the link <https://nanoreview.net/en/soc-list/rating> and finish the table below.

Core-Name	Peak frequency of the most performant block of cores (MHz)			
	Cortex-X3	Cortex-A715	Cortex-A710	Cortex-A510
Snapdragon 8 Gen 2				

Core-Name	Peak frequency of the most performant block of cores (MHz)	
	Everest	Sawtooth
Apple A16 Bionic		

- (b) (5 points) Joe is a wealthy student and uses both his Samsung Galaxy S23 Plus and Apple iPhone 14 Pro Max to run a speech recognition program, such as “Ok Google” and “Hey Siri”, with an equal number of 1742722 words. How long would it take for the program to complete on both devices?

Speech Recognition	Seconds
Samsung	
Apple	

- (c) (5 points) Given that the program in (b) runs on a single core only, specifically the Snapdragon Cortex-X3 for Samsung and Everest for Apple, and there are no other overheads, how many clock cycles does each core take to run the program on both mobile phones?

Speech Recognition	Clock cycles (Million)
Samsung	
Apple	

**Note:**

1.  $CPU\ Time = CPU\ Clock\ Cycles * Clock\ Cycle\ Time = CPU\ Clock\ Cycles / Clock\ Rate$
2. These calculations assume that the frequency of the selected core for running the program is at its highest value.

(d) (5 points) Based on (a) and (c), Joe has executed a self-defined program and obtained the CPI for both of the mobile phones as presented below. Please fill the table below. According to Avg. Ins. time (high freq.), which processor is faster?

Self-defined program	CPI	Highest frequency (MHz) (Cortex-X3, Everest)	Lowest frequency (MHz) (Cortex-A510, Sawtooth)	Avg. Ins. time (high freq.) (10 <sup>-6</sup> sec)	Avg. Ins. time (low freq.) (10 <sup>-6</sup> sec)
Samsung	0.9				
Apple	3.1				
<b>Note</b>					
1. Avg ins. time = Clock cycle time * Avg. Clock per instruction (CPI) = CPI / Clock rate					
2. The gray-colored blanks are not included in the score calculation. They are only provided to expedite the calculation process.					
3. Calculations need to be performed considering the scenario of running on a single core only.					

(e) (5 points) Using the benchmarks of image compression and speech recognition provided in the results table above, calculate the relative performance of the two mobile phones with each phone as the reference for comparison. Complete the following table and summarize the performance results by calculating the arithmetic mean of the performance ratios of the two benchmark programs. Please show the calculation procedure. Based on the results, comment on using the arithmetic mean as performance ratio.

Reference	Performance ratio	
	Samsung	Apple
Samsung	1	

**Note:**

1.  $Performance\ ratio = \frac{the\ execution\ time\ of\ the\ reference\ machine}{the\ execution\ time\ of\ the\ test\ machine}$   
You should solve this problem according to this formula.
2. Assume the workloads are the same for both mobile phones for each of the two tasks, image compression and speech recognition, e.g., both phones have an image of 8k for the image compression task.

3. (15 points)

Here is the definition of the Amdahl's Law.

$$S_{latency} = \frac{1}{(1 - p) + \frac{p}{s}}$$

where

$S_{latency}$  is the theoretical speedup of the execution of the whole task,

$s$  is the speedup of the part of the task that benefits from improved system resources,

$p$  is the proportion of the execution time that the part benefiting from improved resources

originally occupied.

Given a program composed of 4 classes of instructions, in which 10% are class A, 25% class B, 45% class C, and 20% class D, answer the followings, supposing the CPI of each class is 2, 1, 3, and 3.

- (a) (5 points) Based on the eight great ideas of computer architecture introduced in Section 1.2 of the textbook, which class of instructions you would want to improve first and why?
  - (b) (10 points) According to Amdahl's law, what is the speedup if the CPI of classes C and D are both improved to 2? Please show the calculation procedure.
4. (10 points) Assume that a program requires the execution of  $75 \times 10^6$  FP instructions,  $120 \times 10^6$  INT instructions,  $70 \times 10^6$  L/S instructions, and  $45 \times 10^6$  branch instructions. The CPI for each type of instruction is 4, 3, 5, and 1, respectively. Assume that the processor has a 4GHz clock rate.
- (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.
  - (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 29% and the CPI of L/S and Branch is reduced by 56%? Please show the calculation procedure.