

**Department of Computer Science, National Tsing Hua University**  
**CS4100 Computer Architecture**  
Spring 2025, Homework 1  
Due Date: 15<sup>th</sup> March 2025 23:59

1. (48%) **AndeSight**

Installing and using AndeSight™ for RISC-V program development.

(1) See **AndeSight\_STD\_v5.3\_Installation\_Guide.pdf** for the installation guide.

(2) Create a new Andes C Project

(i) Click on **File > New > Project > C/C++**, and select **Andes C Project**.

(ii) Create a project with the project name **hw1\_fibonacci**

(iii) From Chip Profile, select chip profile **AE350 > ADP-AE350-NX25F**

(iv) From Project Type, select project type **Andes Executable > Hello World ANSI C Project**

(v) From Toolchains, select the **nds64le-elf-mculib-v5d**

(vi) Other configurations are left as default

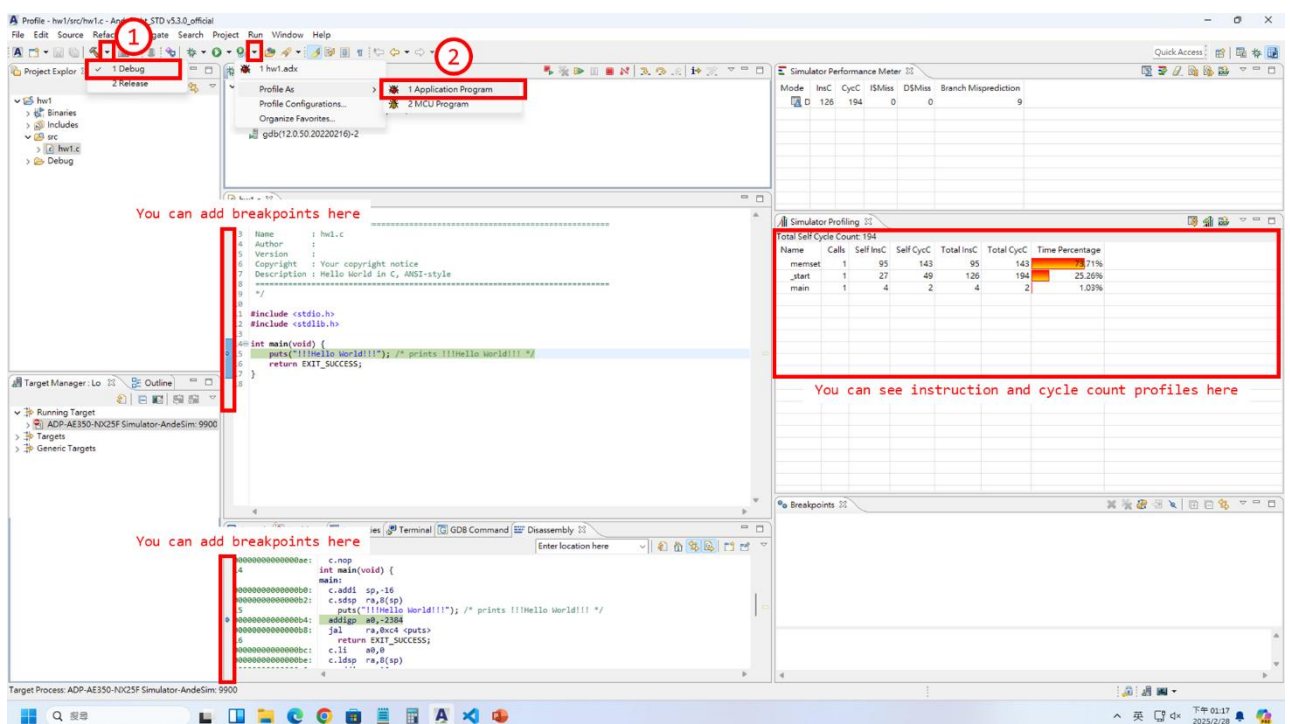
(3) Replace **hw1\_fibonacci.c** in the project with the one we provided.

(4) To build the project, click on the expanding arrow (a small triangle ▾) beside **Build** in the toolbar, then **1 Debug** for project **hw1\_fibonacci** in the toolbar.

(5) To execute the program, press **Debug** in the toolbar, then press **1 Application Program**, and press **Resume** in the debug window.

You can follow the same steps for the other program codes.



To profile the program, follow **Figure 1** below.



**Figure 1.** The steps to profile the selected C program.

In this exercise, we will experiment with the iterative and recursive implementations of the Fibonacci function with AndeSight™. There are two source codes, namely **iter\_fib.c** and **recur\_fib.c**. Unless stated otherwise, the optimization level **-Og** will be used by default in the following questions.

(a) (10%) *Effects of algorithms on performance*

Press **Profile**  in the toolbar and select Profile as **Application Program**. Press the button **Resume**  in the debug window, record the CycC and InsC for the two **functions** listed in the table below and complete **Table 1**. Based on the characteristics of the programs, briefly compare and explain the differences between the iterative and recursive algorithms in their profiles. **Note: When profiling for (a), do not add breakpoints in the execution flow.**

**Table 1.** The cycle count (CycC) and instruction count (InsC) of the two Fibonacci functions.

Function	Source	CycC	InsC
<b>iter_fibonacci()</b>	<b>iter_fib.c</b>		
<b>recur_fibonacci()</b>	<b>recur_fib.c</b>		

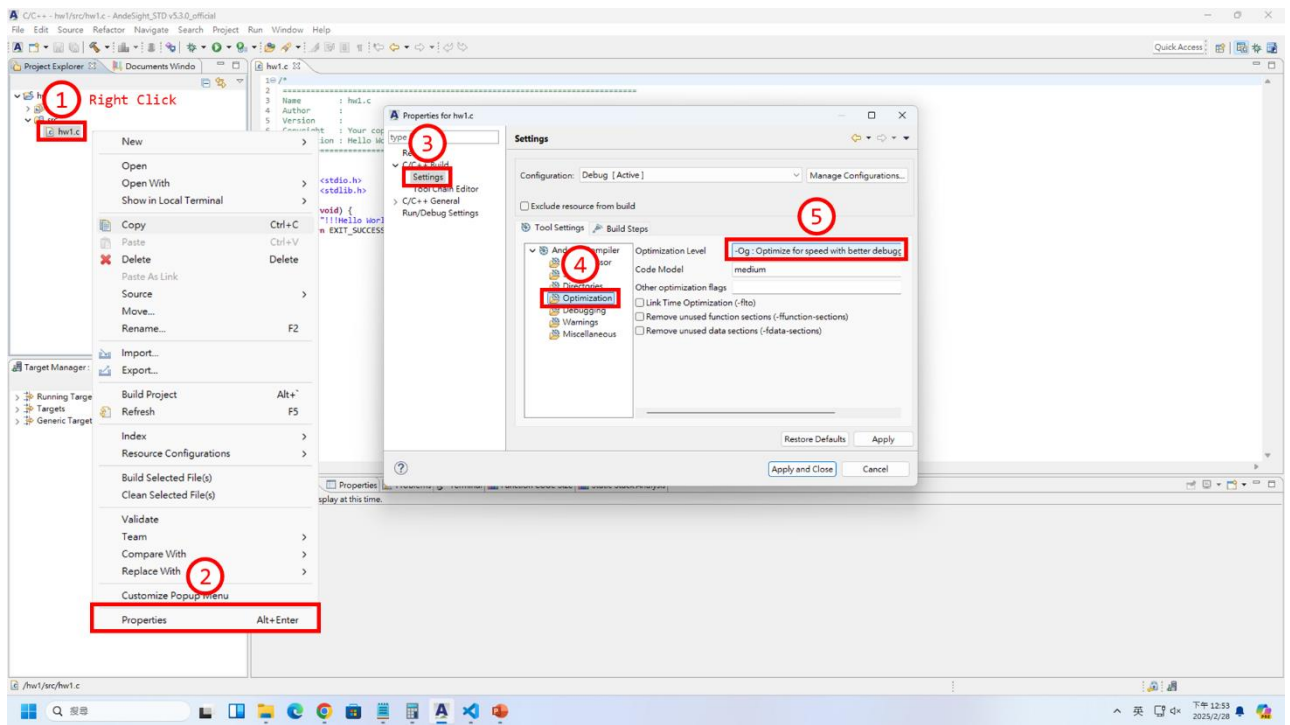
(b) (8%) *Effects of programming on performance*

Given that the programs are executed in a processor with a clock rate of 4 GHz, what are the average CPI and CPU execution time for the **iter\_fibonacci()** and **recur\_fibonacci()** functions? Complete **Table 2**.

**Table 2.** The CPI and CPU Execution times of the two Fibonacci functions.

Function	CPI	CPU Execution Time
<b>iter_fibonacci()</b>		
<b>recur_fibonacci()</b>		

To change (or check) the optimization setting, follow **Figure 2** below.



**Figure 2.** The steps to configure compiler optimization.

(c) (14%) *Effects of the compiler on performance*

Compile **iter\_fib.c** and **recur\_fib.c** with two optimization levels, **-O0** and **-O1**. Record the CycC and InsC and compute their corresponding CPI for the two different optimization levels and complete **Table 3**. Furthermore, briefly compare and explain the differences in their profiles. **Note:** When profiling for (c), **do not** add breakpoints in the execution flow.





**Table 3.** Comparison of CPI between two optimization levels **-O0** and **-O1**.

Function	Optimization level <b>-O0</b>			Optimization level <b>-O1</b>		
	CycC	InsC	CPI	CycC	InsC	CPI
<b>iter_fibonacci()</b>						
<b>recur_fibonacci()</b>						

(d) (8%) **AndeSight IDE**

This section will guide you in using AndeSight™ IDE and breakpoints. Open **iter\_fib.c** in the AndeSight™ IDE and place two breakpoints in the C program:

- One breakpoint at the **for**-loop in **iter\_fibonacci()**, and
- One breakpoint at the **printf()** in **main()**.

Press **Profile**  in the toolbar and select Profile as **Application Program**. Press the button **Resume**  in the debug window, this will jump to the beginning of **main()**. You will need to press **Resume**  button to skip ahead to the next breakpoint encountered in the execution flow of the program. The number of cycle and instruction counts elapsed from the previous breakpoint will be recorded in the latest entry in the **Simulator Performance Meter** .

We assume that we can approximate the cycle count and instruction count in each iteration of the **for**-loop using these breakpoints.

- (4%) Derive the total cycle count (CycC) and instruction count (InsC) for the **for**-loop in **iter\_fibonacci()** by completing Table 4. **Hint:** you can hover your cursor on any variable to check its current value.

**Table 4.** Cycle and instruction counts of the **for**-loop in **iter\_fibonacci()**.

Iteration	CycC	InsC
1		
2		
3		
4		
5		
Total		

- (4%) Using your results of (a), what percentage of the clock cycles do the **for**-loop account in the **iter\_fibonacci()** function?

(e) (8%) **Amdahl's Law**

One of the great ideas of computer architecture is *parallelization*. Amdahl's law can be used to calculate the overall speedup of parallel executions. Amdahl's Law is defined as follows:

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

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, and  $p$  is the proportion of the execution time that the part benefiting from the improved resources originally occupied.

Suppose that you were able to reduce the clock cycles of the **for**-loop by half.

- (4%) Using the results of (d), what is the speedup of the optimized **iter\_fibonacci()** function?
- (4%) Based on the CPU execution time in (b), what is the new CPU execution time?

2. (22%) **Benchmarking**

**Table 5** enlists four smartphones for Questions 2(a), 2(b), 2(c), and 2(d). Their core peak frequencies and the processing speeds of HDR, Ray Tracing, PDF Rendering, and Background Blur are enlisted. Some values are omitted, and you will be asked to calculate the omitted values.

**Table 5.** Specification comparison between Three Smartphones with Single Core Processors.

Smartphone	W	X	Y	Z
Core Peak Frequency	3000 MHz	3200 MHz	2500 MHz	2750 MHz
HDR	600 Mpxs/sec	550 Mpxs/sec	475.2 Mpxs/sec	400 Mpxs/sec
Ray Tracing	12.5 Mpxs/sec	27.5 Mpxs/sec	22 Mpxs/sec	$F$ Mpxs/sec
PDF Rendering	$A$ Mpxs/sec	$B$ Mpxs/sec	$C$ Mpxs/sec	$G$ Mpxs/sec
Background Blur	30 Images/sec	50 Images/sec	12.5 Images/sec	$H$ Images/sec

**Note:** Mpxs/sec is Megapixel per second, for brevity.

In the following questions, the performance ratio of a *query* machine A to a *reference* machine B on Benchmark  $i$  is defined as:

$$\text{Performance Ratio of A to B of Benchmark } i = \frac{\text{Execution Time of B on Benchmark } i}{\text{Execution Time of A on Benchmark } i}$$

(a) (5%) We run three programs 1, 2, and 3 on smartphones W, X, and Y.

**Program 1:** Combine two 4K Resolution images with 16,588,800 pixels using HDR.

**Program 2:** Traces 912,600 rays for rendering a scene in 1280 x 720 HD Resolution.

**Program 3:** Renders 44,000,000 pixels when viewing **HW1\_2025.pdf**.

**Table 6.** Geometric Mean of Performance Ratio of W, X, and Y on Programs 1, 2, and 3.

Query \ Reference	W	X	Y
W	1.000	1.100	0.880
X	0.909	1.000	0.800
Y	$D$	$E$	1.000

Suppose that the geometric mean of the performance ratio is given by **Table 6**. Given that smartphone Y completes Program 3 in 0.1 seconds, what are the values  $A$ ,  $B$ ,  $C$ ,  $D$ , and  $E$ ?

(b) (5%) Suppose that we run programs 2, 3, and 4 on smartphones X, Y, and Z.

**Program 2:** Traces 912,600 rays for rendering a scene in 1280 x 720 HD Resolution.

**Program 3:** Renders 44,000,000 pixels when viewing **HW1\_2025.pdf**.

**Program 4:** Blurs the backgrounds of 125,000 images in an image classification dataset.

**Table 7.** Arithmetic Mean of Performance Ratio of X, Y, and Z on Programs 2, 3, and 4.

Reference \ Query	X	Y	Z
X	1.000	0.597	0.800
Y	2.200	1.000	1.855
Z	<i>I</i>	<i>J</i>	1.000

Suppose that the arithmetic mean of the performance ratio is given by **Table 7**. Using information from (a) and given that *G* is 90% of *B*, what are the values *F*, *G*, *H*, *I*, and *J*?

(c) (6%) Using information from (a) and (b), by what factor should we increase the PDF Render processing speed of smartphone Z to surpass smartphones W, X, and Y in geometric mean of performance ratios on Programs 1, 2, 3, and 4? Provide your reasons and calculation procedures.

(d) (6%) Using information from (a) and (b), what values of *H* will cause smartphone Z to simultaneously satisfy the three conditions below? Provide your reasons and calculation procedures.

- 1) Smartphone Z beats all smartphones W, X, and Y in terms of geometric mean of performance ratio on Programs 1, 2, 3, and 4.
- 2) Smartphone Z is inferior to smartphone X in the arithmetic mean of performance ratio on Programs 1, 2, 3, and 4 when taking smartphone X as reference.
- 3) Smartphone Z is inferior to smartphone Y in the arithmetic mean of performance ratio on Programs 1, 2, 3, and 4 when taking smartphone Y as reference.

3. (10%) **Performance and Speedup**

Assume that a program requires the execution of  $150 \times 10^6$  FP instructions,  $60 \times 10^6$  INT instructions,  $75 \times 10^6$  L/S instructions, and  $25 \times 10^6$  branch instructions. We have two processors, P1 and P2, which have different CPI values for each instruction type:

**Table 8.** Instruction Type and CPI of The Two Processors P1 and P2

Instruction Type	CPI on P1	CPI on P2
Floating Point (FP)	2	1
Integer (INT)	3	2
Load/Store (L/S)	3	2
Branch	2	2

Suppose that each instruction type with their corresponding CPI on the 2 different processors are given in **Table 8**. Given that Processor P1 runs on 4 GHz and Processor P2 runs on 3 GHz:

- (5%) A common fallacy is that a processor with a higher clock speed always has better performance. Is P1 actually faster than P2? Justify your answer with calculations.
- (5%) By how much is the execution time of the program in processor P1 and P2 improved if the CPI of INT and FP instructions is reduced by 40% and the CPI of L/S and Branch is reduced by 25%? Show the calculation procedure.

4. (20%) **Amdahl's Law and the Eight Great Ideas of Computer Architecture**

Given the definition of Amdahl's Law as

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

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, and  $p$  is the proportion of the execution time that the part benefiting from the improved resources originally occupied.

Given a program composed of 4 classes of instructions, in which 55% are class A, 15% class B, 20% class C, and 10% class D, answer the following, supposing the CPI of each class is 3, 2, 2, and 3.

- (5%) Based on the "Eight Great Ideas of Computer Architecture" from Section 1.2 of the textbook, which instruction class would you prioritize for improvement and why?
- (5%) According to Amdahl's Law, what would be the speedup if the CPI of the class you chose in (a) is improved to 1.5? Show the calculation procedure.
- (5%) According to Amdahl's Law, what would be the speedup if the CPI of classes C and D were both improved to 1? Show the calculation procedure.
- (5%) Currently, you want to achieve a 2x speedup of the original program by improving only the CPI of class B. What should the new CPI of class B be to achieve this? Show the calculation procedure.