EECS403000 Computer Architecture
Spring, 2023 Homework 1

(1)、

(a) **(6 points)**

iter: InsC: 46, CycC: 85

recur: InsC: 136, CycC: 245

Recursive execution to store additional register value into the stack(more instructions), thus introducing additional time to access memory. The execution time will rise in this example.

(b) **(6 points)**

iter:

CPI = Cycle count / Instruction count

Substituting the given values, we get: 85 / 46 ≒ 1.85

To calculate the execution time of a program with the given information and a clock rate of 2 GHz, we can use the following formula:

Execution time = Cycle count / Clock rate

The clock rate is given as 2 GHz, which means the clock cycle time is 1/2 GHz = 0.5 nanoseconds.

Substituting the values in the formula, we get: 85 * 0.5ns = 42.5ns

recur:

245 / 136 ≒ 1.80

245 * 0.5ns = 122.5ns

(c) (6 points)

a0, a0

(d) **(10 points)**

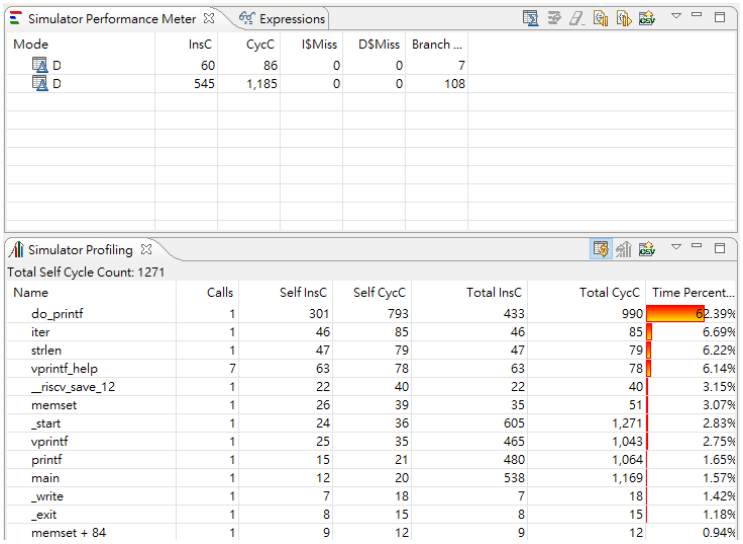
optimization levels, -O0		optimization levels, -O1	
For Factorial_iter	For Factorial_recur	For Factorial_iter	For Factorial_recur
InsC: 150	InsC: 242	InsC: 36	InsC: 145
CycC: 287	CycC: 411	CycC: 69	CycC: 222
CPI: 1.913	CPI: 1.698	CPI: 1.917	CPI: 1.531

-O0: Do not optimize; -O1: Optimize for speed. -Og: Optimize for speed with better debuggability than -O1

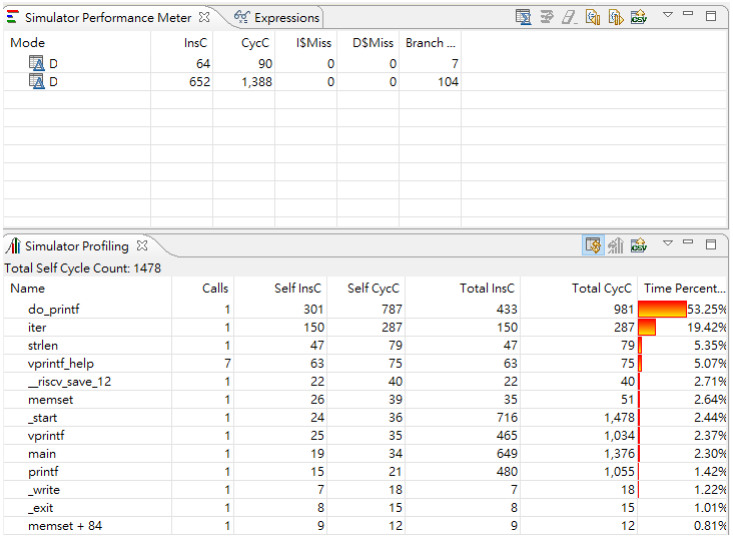
Compared with -O0, -O1 decreased InsC and CycC.

Details about GCC optimization options:Options That Control Optimization

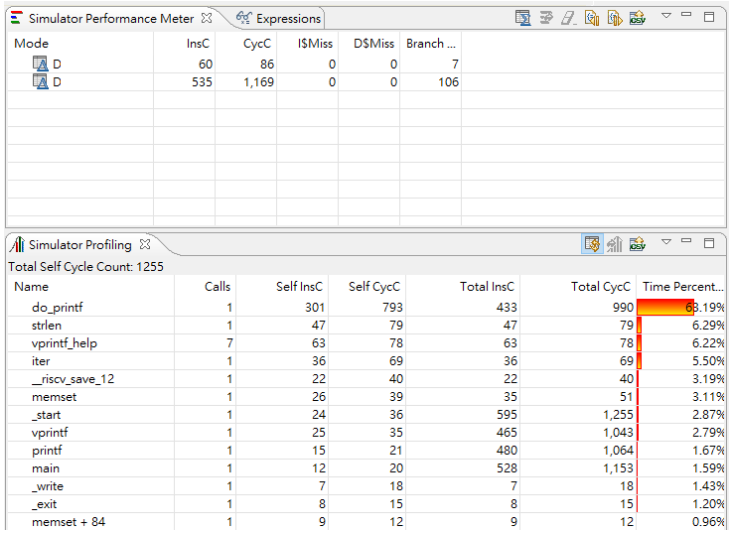
1. Factorial_iterative
   A. Opt: -Og



B. Opt: -O0

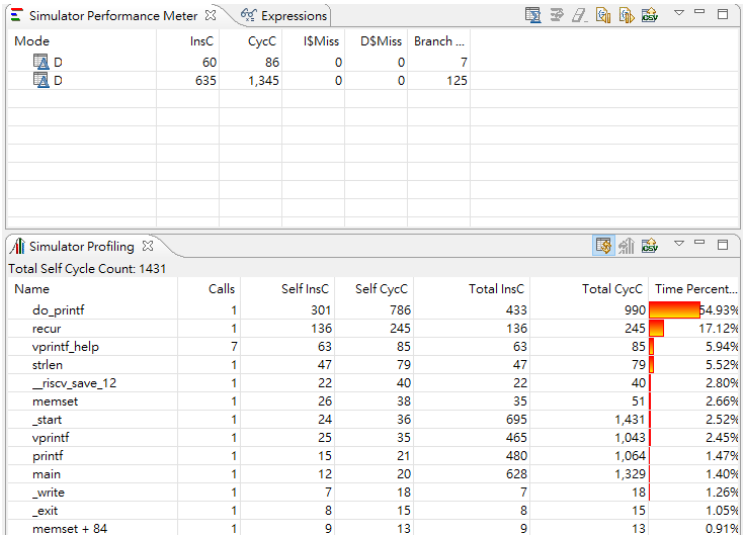


C. Opt: -O1



2. Factorial_recursive

A. Opt: -Og



Simulator Performance Meter | Expressions

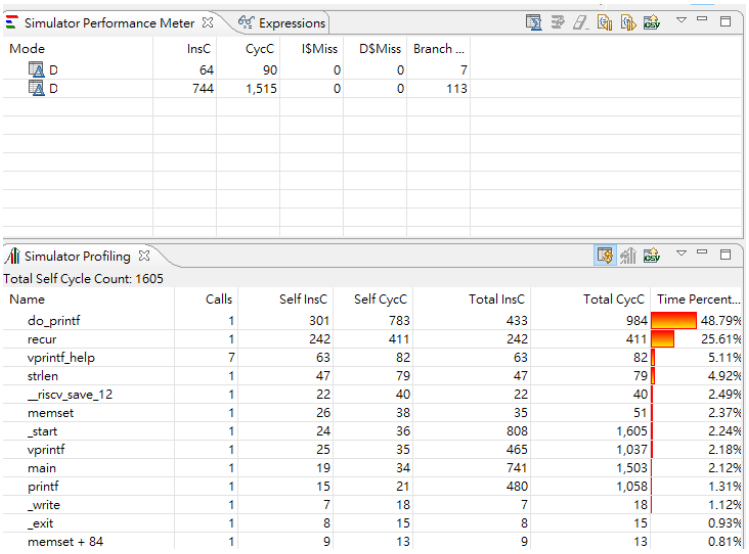
Mode	InsC	CycC	I$Miss	D$Miss	Branch ...
D	60	86	0	0	7
D	635	1,345	0	0	125

Simulator Profiling

Total Self Cycle Count: 1431

Name	Calls	Self InsC	Self CycC	Total InsC	Total CycC	Time Percent...
do_printf	1	301	786	433	990	54.93%
recur	1	136	245	136	245	17.12%
vprintf_help	7	63	85	63	85	5.94%
strlen	1	47	79	47	79	5.52%
__riscv_save_12	1	22	40	22	40	2.80%
memset	1	26	38	35	51	2.66%
_start	1	24	36	695	1,431	2.52%
vprintf	1	25	35	465	1,043	2.45%
printf	1	15	21	480	1,064	1.47%
main	1	12	20	628	1,329	1.40%
_write	1	7	18	7	18	1.26%
_exit	1	8	15	8	15	1.05%
memset + 84	1	9	13	9	13	0.91%

B. Opt: -O0



Simulator Performance Meter | Expressions

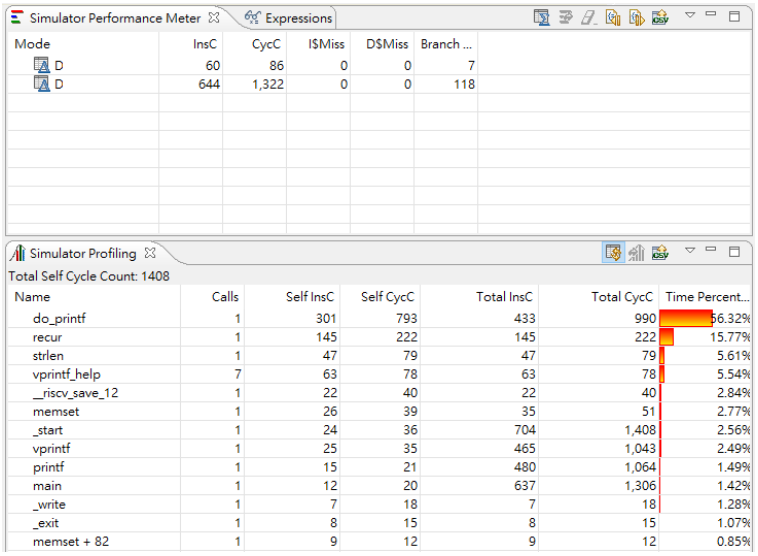
Mode	InsC	CycC	I$Miss	D$Miss	Branch ...
D	64	90	0	0	7
D	744	1,515	0	0	113

Simulator Profiling

Total Self Cycle Count: 1605

Name	Calls	Self InsC	Self CycC	Total InsC	Total CycC	Time Percent...
do_printf	1	301	783	433	984	48.79%
recur	1	242	411	242	411	25.61%
vprintf_help	7	63	82	63	82	5.11%
strlen	1	47	79	47	79	4.92%
__riscv_save_12	1	22	40	22	40	2.49%
memset	1	26	38	35	51	2.37%
_start	1	24	36	808	1,605	2.24%
vprintf	1	25	35	465	1,037	2.18%
main	1	19	34	741	1,503	2.12%
printf	1	15	21	480	1,058	1.31%
_write	1	7	18	7	18	1.12%
_exit	1	8	15	8	15	0.93%
memset + 84	1	9	13	9	13	0.81%

C. Opt: -O1



Simulator Performance Meter | Expressions

Mode	InsC	CycC	I$Miss	D$Miss	Branch ...
D	60	86	0	0	7
D	644	1,322	0	0	118

Simulator Profiling

Total Self Cycle Count: 1408

Name	Calls	Self InsC	Self CycC	Total InsC	Total CycC	Time Percent...
do_printf	1	301	793	433	990	56.32%
recur	1	145	222	145	222	15.77%
strlen	1	47	79	47	79	5.61%
vprintf_help	7	63	78	63	78	5.54%
__riscv_save_12	1	22	40	22	40	2.84%
memset	1	26	39	35	51	2.77%
_start	1	24	36	704	1,408	2.56%
vprintf	1	25	35	465	1,043	2.49%
printf	1	15	21	480	1,064	1.49%
main	1	12	20	637	1,306	1.42%
_write	1	7	18	7	18	1.28%
_exit	1	8	15	8	15	1.07%
memset + 82	1	9	12	9	12	0.85%

(e) (12 points)

i. **(4 points)**

CISC vs RISC:

- Instruction Set: CISC processors have a large number of complex instructions that can perform multiple operations in a single instruction. RISC processors have a smaller number of simple instructions that perform only one operation in each instruction. This makes RISC instructions simpler and faster to execute, but CISC instructions can sometimes be more efficient in terms of code size.
- Hardware Complexity: CISC processors are generally more complex than RISC processors, because they have to support a larger number of instructions. This means that CISC processors require more transistors and other hardware components than RISC processors, which can make them more expensive and harder to design.
- Pipeline Efficiency: RISC processors are designed to execute instructions in a pipeline, which means that they can start executing the next instruction before the current instruction has finished. This can make RISC processors more efficient than CISC processors in terms of execution time.
- Memory Usage: CISC processors typically require more memory than RISC processors, because their instructions are larger and more complex. This can make CISC processors less efficient in memory-constrained systems.
- Power Consumption: RISC processors are generally more power-efficient than CISC processors, because their simpler instructions require less power to execute. This makes RISC processors more suitable for mobile and battery-powered devices.

ii. **(4 points)**

	IC	CPI	Cycle time
RISC	more	less	same
CISC	less	more	

iii. **(4 points)**

0.65*1.25 = 0.8125<1

or

40%*0.65 * (30%/40%) * 1.25 = 24.375% < 30%(cycles)

(f) **(10 points)**

i、 **(4 points)**

Pipeline:(T refers to time)

```
T1      T2      T3      T3      T4      T5      T6
i++;    sum *= i;
        i++;    sum *= i;
                i++;    sum *= i;
                        i++;    sum *= i;
                                i++;    sum *= i;
```

i++; sum *= i;

i++ => 1 cycle
sum *= i => 1 cycle

Although the number of cycles is equal, the pipeline will always have n+1 cycles, where n = total cycles(instructions)/stages. Here, we can only reach at most speedup ≈ 2.

instruction obs on simulator performance meter view:
mulw a4,a4,a5 : 1 cyc
c.addiw a5,1 : 1 cyc
c.nop : 1 cyc
bge a0,a5,0x104b0 <iter+8>: 4 cycs

ii、 **(6 points)**

Q1:

Execute a for loop: 7 cycles, so the cycles except for loop in iter() are:
85-7*10 = 15
3 stages procedure:

1. Let Process A do 1 * 1 * 2, B do 3 * 4 * 5, C do 6 * 7 * 8, D do 9 * 10
   - takes 2 multiplications(loops), total 2*7 = 14 cycles
2. Process B do result A * result B, D do result C * result D
   - takes 1 multiplication(loop), total 1*7 = 7 cycles
3. Process D do result B * result D
   - takes 1 multiplication(loop), total of 1*7 = 7 cycles

Parallel takes: 15 + 14 + 7 + 7 = 43 cycles
=> reduced 42 cycles
=> percentage reduction 42/85 = 49%

Q2:

No, iter() function has instructions other than computation, since the result of computation needs to be merged the speedup of 4 cannot be achieved.

(2)、 **(25 points)**

(a) Follow the link https://nanoreview.net/en/soc-list/rating , try to 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	3200	2800	2800	2000
Core-Name	Peak frequency of the most performant block of cores (MHz)			

	Everest	Sawtooth		
Apple A16 Bionic	3460	2020		

(b) Assuming Joe is a wealthy student, he 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	14180
Apple	12290

(c) Given that the program described in question (b) runs on a single core only, specifically the Snapdragon Coretex-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	Seconds	Frequency	Clocks (Million)
Samsung	14180	3200	45376000
Apple	12290	3460	42523400

(d) 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	The highest frequency MHz	The lowest frequency MHz	Avg. Ins. time (high freq.) (10^-6s)	Avg. Ins. time (low freq.) (10^-6)
Samsung	0.9	3200	2000	0.00028125	0.00045
Apple	3.1	3460	2020	0.000895953757 2	0.001534653465

(e) Using the benchmarks of image compression and speech recognition provided in the results table, 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.

Reference	Performance ratio	
	Samsung	Apple
Samsung	1	1.0613

calculating process:

Task	Samsung	Apple
Image compression performance	251.3	243.5
Speech recognition performance	122.9	141.8
performance ratio	1	$(\frac{243.5}{251.3}+\frac{141.8}{122.9})\frac{1}{2} = 1.0613...$

(3)、

(a) According to the eight great ideas of computer architecture: Make the Common Case Fast

According to textbook 1.2 'Make the Common Case Fast,' it is common practice to optimize the most frequent case in order to speed up the overall performance. Therefore, we would like to prioritize optimizing the instructions of class C, which have the highest proportion and are the most common in the entire program.

(b)

$$p=\frac{0.45*3+0.2*3}{0.1*2+0.25*1+0.45*3+0.2*3}=\frac{1.95}{2.4}=0.8125$$

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

$$=\frac{1}{(1-0.8125)+0.8125/1.5}$$

$$=1.371429$$

(4)、

(a)

$$Clock\ cycles = CPI_{fp}\times\#\ of\ FP\ instr. + CPI_{int}\times\#\ of\ INT\ instr. + CPI_{l/s}\times\#\ of\ L/S\ instr. + CPI_{branch}\times\#\ of\ BRANCH\ instr.$$

$$T_{CPU} = clock\ cycles\ /\ clock\ rates$$

$$Clock\ cycles = (4\times75 + 3\times120 + 5\times70 + 1\times45)\times10^6 = 1055\times10^6$$

$$\frac{T_{CPU\ new}}{T_{CPU\ old}} = \frac{clock\ cycles_{new}}{clock\ cycles_{old}} = \frac{1}{2}$$

To have the number of clock cycles by improving the CPI of FP instructions:

$$CPI_{fp\ improved}\times\#\ of\ FP\ instr. + CPI_{int}\times\#\ of\ INT\ instr. + CPI_{l/s}\times\#\ of\ L/S\ instr. + CPI_{branch}\times\#\ of\ BRANCH\ instr. = clock\ cycles\ /\ 2$$

$$CPI_{fp\ improved} = (clock\ cycles\ /\ 2\ -\ (CPI_{int} \times \#\ of\ INT\ instr.\ +\ CPI_{l/s} \times \#\ of\ L/S\ instr.\ +\ CPI_{branch} \times \#\ of\ BRANCH\ instr.\ ))\ /\ \#\ of\ FP\ instr.$$

$$CPI_{fp\ improved} = \frac{527.5-755}{75} < 0$$

 impossible

(b)

$$\frac{T_{CPU\ new}}{T_{CPU\ old}} = \frac{clock\ cycles_{new}}{clock\ cycles_{old}} = \frac{4\times75\times0.71+3\times120\times0.71+5\times70\times0.44+1\times45\times0.44}{4\times75+3\times120+5\times70+1\times45} = 0.61$$

 reduced by 39%