Computer Architecture HW 1

111062117, Hsiang-Sheng Huang

March 13, 2025

1

(a)

Function	source	CycC	InsC
iter_fibonacci()	iter_fib.c	57	36
recur_fibonacci()	recur_fib.c	407	293

The recursive function **recur_fibonacci()** runs in $O(\phi^n)$ time complexity, where ϕ is the golden ratio. The iterative function **iter_fibonacci()** runs in $O(n)$ time complexity. The number of cycles and instructions for the recursive function is much larger than the iterative function, which is consistent with the time complexity analysis.

(b)

$$\text{CPI} = \frac{\text{CycC}}{\text{InsC}}$$

$$\text{CPU Execution Time} = \frac{\text{CycC}}{\text{Clock Rate}}$$

The formula for calculating **CPI** and **CPU Execution Time** is shown above. And the clock rate is 4GHz given in the question. Following the formula, we can calculate the **CPI** and **CPU Execution Time** for both functions.

Function	CPI	CPU Execution Time
iter_fibonacci()	1.5833	14.25ns
rec_fibonacci()	1.3891	101.75ns

(c)

Function	Optimization level -O0			Optimization level -O1		
	CycC	InsC	CPI	CycC	InsC	CPI
iter_fibonacci()	278	110	2.53	52	34	1.53
recur_fibonacci()	1327	823	1.61	404	293	1.38

- O0: No optimization.
- O1: Optimization for speed.
- Og: Optimization for speed with better debuggability than O1.

It uses less CycC and InsC to run the program with optimization level -O1 than -O0.

(d)

(i)

Iteration	CycC	InsC
1	8	6
2	8	6
3	8	6
4	8	6
5	8	6
Total	40	30

(ii)

We know that the cycles for **iter_fibonacci()** is 57 and **for**-loop account for 40 cycles. Therefore, the percentage of cycles spent on the **for**-loop is:

$$\frac{40}{57} \times 100\% \approx 70.18\%$$

(e)

(i)

$$s = 2, \quad p = \frac{40}{57} \approx 0.7018$$

So the overall speedup is:

$$S_{\text{latency}} = \frac{1}{(1-p) + \frac{p}{s}} = \frac{1}{(1-0.7018) + \frac{0.7018}{2}} = \frac{1}{0.2982 + 0.3509} = \frac{1}{0.6491} \approx 1.54.$$

(ii)

The new CPU execution time is calculated by dividing the original CPU execution time by the overall speedup. Thus,

$$\text{New CPU Execution Time} = \frac{14.25\,\text{ns}}{1.54} \approx 9.25\,\text{ns}.$$

2

(a)

Since we use the geometric mean to calculate the performance ratios, the product of the two relative performance ratios is 1, meaning that each device's performance is the reciprocal of the other's. Thus we have:

$$0.880 \times D = 1 \quad \Rightarrow \quad D = \frac{1}{0.880} \approx 1.136,$$

$$0.800 \times E = 1 \quad \Rightarrow \quad E = \frac{1}{0.800} = 1.25.$$

Then, for Program 3 (PDF Rendering), we know that 44,000,000 pixels are rendered and smartphone Y completes the task in 0.1 seconds. Therefore, Y's PDF rendering speed is:

$$C = \frac{44000000}{0.1} = 440 \text{ Mpxs/sec}.$$

Then, we use the geometric mean to calculate the performance ratios for smartphones W and Y:

$$\sqrt[3]{\frac{600}{475.2} \times \frac{12.5}{22} \times \frac{\mathrm{A}}{440}} = 1.136.$$
$$A \approx 899$$

Similarly, we calculate the performance ratios for smartphones X and Y:

$$\sqrt[3]{\frac{550}{475.2} \times \frac{27.5}{22} \times \frac{\mathrm{B}}{440}} = 1.25.$$
$$B \approx 594$$

In summary, the values are:

$$A \approx 899, \quad B \approx 594, \quad C = 440, \quad D \approx 1.136, \quad E = 1.25.$$

(b)

From (a), we have:

$$B = 594 \quad \Longrightarrow \quad G = 0.9 \times B = 534.6,$$

Then, use the arithmetic-mean entries in Table 7 to set up linear equations for each cell.
We set X as reference and Z as query, and we have:

$$\frac{1}{3}\left(\frac{F}{27.5} + \frac{534.6}{594} + \frac{H}{50}\right) = 0.800$$

We set Y as reference and Z as query, and we have:

$$\frac{1}{3}\left(\frac{F}{22} + \frac{534.6}{440} + \frac{H}{12.5}\right) = 1.855$$

Solving these two equations, we get: $F = 16.5$ and $H = 45$.
To calculate the value of I, we set Z as reference and X as query, and we have:

$$\frac{1}{3}\left(\frac{27.5}{16.5} + \frac{594}{534.6} + \frac{50}{45}\right) \approx 1.296$$

To calculate the value of J, we set Z as reference and Y as query, and we have:

$$\frac{1}{3}\left(\frac{22}{16.5} + \frac{440}{534.6} + \frac{12.5}{45}\right) \approx 0.811$$

In summary, the values are:

$$F = 16.5, \quad H = 45, \quad G = 534.6, \quad I \approx 1.296, \quad J \approx 0.811.$$

(c)

$$\mathrm{performance_W} = \sqrt[4]{600 \times 12.5 \times 900 \times 30} \approx 119.3$$
$$\mathrm{performance_X} = \sqrt[4]{550 \times 27.5 \times 594 \times 50} \approx 145.6$$
$$\mathrm{performance_Y} = \sqrt[4]{475.2 \times 22 \times 440 \times 12.5} \approx 87.1$$
$$\mathrm{performance_Z} = \sqrt[4]{400 \times 16.5 \times G' \times 45} > \max(\mathrm{performance_W}, \mathrm{performance_X}, \mathrm{performance_Y}) \approx 145.6$$
$$G' > \frac{145.6^4}{400 \times 16.5 \times 45} \approx 1512.8.$$

So the speedup factor would be:

$$\frac{1512.8}{534.6} \approx 2.83.$$

(d)

1)

Similar to (c), we have:

$$\text{performance}_\text{Z} = \sqrt[4]{400 \times 16.5 \times 534.6 \times H'} > \max(\text{performance}_\text{W}, \text{performance}_\text{X}, \text{performance}_\text{Y}) \approx 145.6$$

$$H' > \frac{145.6^4}{400 \times 16.5 \times 534.6} \approx 127.3.$$

2)

$$\frac{1}{4}\left(\frac{400}{550} + \frac{16.5}{27.5} + \frac{534.6}{594} + \frac{H'}{50}\right) < 1$$

$$H' < 50 \times \left(4 - \left(\frac{400}{550} + \frac{16.5}{27.5} + \frac{534.6}{594}\right)\right) \approx 88.6.$$

3)

$$\frac{1}{4}\left(\frac{400}{475.2} + \frac{16.5}{22} + \frac{534.6}{440} + \frac{H'}{12.5}\right) < 1$$

$$H' < 12.5 \times \left(4 - \left(\frac{400}{475.2} + \frac{16.5}{22} + \frac{534.6}{440}\right)\right) \approx 14.93.$$

The intersection of the three constraints is:

$$H' > 127.3, \quad H' < 88.6, \quad H' < 14.93 \quad \Longrightarrow \quad \text{No feasible value for } H'.$$

So there is no H' that satisfies all constraints.

3

(a)

By formula,

$$\text{CPU Execution Time} = \frac{\text{CPI} \times \text{instructions}}{\text{Clock Rate}}.$$

We compute the cycles for each processor first:

$$\text{Cycles}_\text{P1} = 2 \times (150 \times 10^6) + 3 \times (60 \times 10^6) + 3 \times (75 \times 10^6) + 2 \times (25 \times 10^6) = 755 \times 10^6.$$

$$\text{Cycles}_\text{P2} = 1 \times (150 \times 10^6) + 2 \times (60 \times 10^6) + 2 \times (75 \times 10^6) + 2 \times (25 \times 10^6) = 470 \times 10^6.$$

Then, we can calculate the CPU execution time for each processor:

$$\text{CPU Execution Time}_\text{P1} = \frac{755 \times 10^6}{4 \times 10^9} = 0.18875 \text{ seconds.}$$

$$\text{CPU Execution Time}_\text{P2} = \frac{470 \times 10^6}{3 \times 10^9} = 0.15667 \text{ seconds.}$$

P1 is not faster than P2.
This result demonstrates that a higher clock speed does not necessarily mean better performance.
The overall execution time depends on both the clock speed and the CPI.

(b)

The new CPI values are:

Instruction Type	CPI on P1	CPI on P2
FP	1.2	0.6
INT	1.8	1.2
L/S	2.25	1.5
Branch	1.5	1.5

Then, the new cycles for each processor are:

$$\text{Cycles}_{\text{P1}} = 1.2 \times (150 \times 10^6) + 1.8 \times (60 \times 10^6) + 2.25 \times (75 \times 10^6) + 1.5 \times (25 \times 10^6) = 494.25 \times 10^6,$$

$$\text{Cycles}_{\text{P2}} = 0.6 \times (150 \times 10^6) + 1.2 \times (60 \times 10^6) + 1.5 \times (75 \times 10^6) + 1.5 \times (25 \times 10^6) = 312 \times 10^6.$$

The new CPU execution times are:

$$\text{CPU Execution Time}_{\text{P1}} = \frac{494.25 \times 10^6}{4 \times 10^9} \approx 0.12356 \text{ seconds},$$

$$\text{CPU Execution Time}_{\text{P2}} = \frac{312 \times 10^6}{3 \times 10^9} \approx 0.104 \text{ seconds}.$$

The improvements are:

$$\text{Improvement}_{\text{P1}} = \frac{0.18875}{0.12356} \approx 1.526,$$

$$\text{Improvement}_{\text{P2}} = \frac{0.15667}{0.104} \approx 1.507.$$

4

(a)

The class A instructions should be prioritized for improvement. Since they account for the majority of the instructions (55%) and have the highest CPI (3), improving the CPI of class A instructions will have the greatest impact on the overall performance of the program.
This matches **"Make the Common Case Fast"** from the "Eight Great Ideas of Computer Architecture".

(b)

Since the CPI for class A is reduced from 3 to 1.5, the speedup for class A is:

$$s = \frac{3}{1.5} = 2.$$

The fraction of total cycles attributable to class A is calculated as:

$$p = \frac{\text{Class A cycles}}{\text{Total cycles}} = \frac{0.55 \times 3}{0.55 \times 3 + 0.15 \times 2 + 0.2 \times 2 + 0.1 \times 3} = \frac{1.65}{2.65} \approx 0.623.$$

Using Amdahl's Law, the overall speedup is:

$$S_{\text{latency}} = \frac{1}{(1-p) + \frac{p}{s}} = \frac{1}{(1-0.623) + \frac{0.623}{2}} = \frac{1}{0.377 + 0.3115} = \frac{1}{0.6885} \approx 1.451.$$

(c)

Original average CPI:

$$\text{CPI}_{\text{avg}} = 0.55 \times 3 + 0.15 \times 2 + 0.2 \times 2 + 0.1 \times 3 = 2.65.$$

If the CPI of classes C and D are both improved to 1, the new average CPI is:

$$\text{CPI}_{\text{avg}} = 0.55 \times 3 + 0.15 \times 2 + 0.2 \times 1 + 0.1 \times 1 = 2.25.$$

The overall speedup is:

$$S_{\text{latency}} = \frac{2.65}{2.25} \approx 1.178.$$

(d)

To achieve a 2x speedup of the original program by improving only the CPI of class B, the total cycle count must be reduced to

$$\frac{2.65}{2} = 1.325.$$

Since the combined cycles for classes A, C, and D is 2.35, we have:

$$2.35 + 0.15 \times \text{CPI}_{\text{new}} = 1.325.$$

Solving for the new CPI for class B:

$$0.15 \times \text{CPI}_{\text{new}} = 1.325 - 2.35 = -1.025,$$

$$\text{CPI}_{\text{new}} = \frac{-1.025}{0.15} \approx -6.83.$$

Therefore, to achieve a 2x speedup by improving only class B, the new CPI would have to be -6.83, which is impossible.