108062213 顏浩昀
Computer Architecture HW1
2022.03.10

1.

a). 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.

	CycC	InsC
add_recur	375	276
add_iter	105	66

As I know, there are at least two reasons causing add_recur function need more Instruction and Cycle. Firstly, the program needs to initiate a new variable "n" when recursive function to be called. In this case, we need to initiate "n" 20 times rather than 1 time if we use add_iter function. The other one is we need to repeat calling add_recur function if we use recursive way. However, all function calls must be stored in a stack to allow the return to the caller functions. It causes a large amount of overhead.

b). RISC-V has 32 general purpose registers, whereas X86 (a CISC architecture) only has 8. Why does RISC have more registers than CISC? When executing "ADD_iterative", which register stores the return value for add_iter function?

In RISC architecture, one complex instruction may be divided into a few simple instructions. So, RISC need more registers. In contrast, CISC may do the complex instruction directly on memory bank and doesn't require programmer to call "loading" or "storing" functions, which means program doesn't need to use lots of registers.

As I tried to find which register store return value, I set break point at line 9 and line 15 individually. I find register a4 store "sum" and finally return its value in add_iter function and register a0 receive the return value of add_iter function.

c). What are the average CPI for "add_recur" function in ADD_recursive.c and "add_iter" function in ADD_iterative.c, respectively?

For add_recur function $CPI = \frac{375}{276} \approx 1.36$

For add_iter function $CPI = \frac{105}{66} \approx 1.59$

d). 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?

For add_recur function $CPU\ exe\ time = \frac{375}{1GHz} = 375ns$

For add_iter function $CPU\ exe\ time = \frac{105}{1GHz} = 105ns$

e). 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?

We distribute 105 cycle of add_iter function to 4 cores, so we need 27 cycles to complete it. The rest of the program need 900-105=795 cycles to finish. Moreover, because we use parallelized technique, we need to spend 50 extra cycle let cores do communication. Hence, we totally need 795+27+50=872 cycles to complete this program.

The answer $Program\ exe\ time = \frac{795+27+50}{1GHz} = 872ns$

f). 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.

	CycC	InsC	CPI
add_recur (-O0)	763	492	1.55
add_recur (-O1)	394	295	1.34
add_iter (-O0)	576	259	2.22
add_iter (-O1)	98	65	1.51

Level O1 let program become smaller and execute faster than O0, and it doesn't take more time to compiler since the complex optimization such as instruction scheduling isn't optimized in this level. As we can see in above table, instruction counts in O1 are significantly less than amount in O0, and CPI also become slightly smaller.

2.

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

	CPU Characteristics	CPU MHz	CPU(s) Enabled	L3 Cache
I3-4340	-	3600	2 cores/chip 2 threads/core 1chip	4MB
I5-4430	Turbo boost up to 3.2GHz	3000	4 cores/chip 1chip	6MB
I7-4770	Turbo boost up to 3.9GHz	3400	4 cores/chip 2 threads/core 1chip	8MB

b). 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.

Round to 2nd decimal place

Reference	Performance ratio of 464.h264ref		
	I3-4340	I5-4430	I7-4770
I3-4340	1	323/373=0.87	323/323=1
I5-4430	373/323=1.16	1	373/323=1.16
I7-4770	323/323=1	323/373=0.87	1

Reference	Performance ratio of 471.omnetpp		
	I3-4340	I5-4430	I7-4770
I3-4340	1	257/251=1.02	257/223=1.15
I5-4430	251/257=0.98	1	251/223=1.13
I7-4770	223/257=0.87	223/251=0.89	1

Reference	Performance ratio of 473.astar		
	I3-4340	I5-4430	I7-4770
I3-4340	1	233/247=0.94	233/200=1.17
I5-4430	247/233=1.06	1	247/200=1.24

I7-4770	200/233=0.86	200/247=0.81	1

Reference	Mean of Performance ratio		
	I3-4340	I5-4430	I7-4770
I3-4340	1	(0.87+1.02+0.94)/3=0.94	(1+1.15+1.17)/3=1.11
I5-4430	(1.16+0.98+1.06)/3=1.07	1	(1.16+1.13+1.24)/3=1.18
I7-4770	(1+0.87+0.86)/3=0.91	(0.87+0.89+0.81)/3=0.86	1

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

Reference	Geometric Mean of Performance ratio		
	I3-4340	I5-4430	I7-4770
I3-4340	1	$\sqrt[3]{0.87 * 1.02 * 0.94} = 0.94$	$\sqrt[3]{1 * 1.15 * 1.17} = 1.10$
I5-4430	$\sqrt[3]{1.16 * 0.98 * 1.06} = 1.06$	1	$\sqrt[3]{1.16 * 1.13 * 1.24} = 1.18$
I7-4770	$\sqrt[3]{1 * 0.87 * 0.86} = 0.91$	$\sqrt[3]{0.87 * 0.89 * 0.81} = 0.86$	1

d). Which observations can you make from (b) and (c)? How are these results compared with SPECint_base2006 ratio?

From above 2 table, we know i7-4770 gets the best performance among them, and i5-4430 gets the worst performance. In this case, geometric mean of performance ratio and mean of performance ratio get almost same result.

Reference	Ratio of SPECint_base2006 ratio		
	I3-4340	I5-4430	I7-4770
I3-4340	1	48.3/52.3=0.92	59.6/52.3=1.14
I5-4430	52.3/48.3=1.08	1	59.6/48.3=1.23
I7-4770	52.3/59.6=0.88	48.3/59.6=0.81	1

As we look at SPECint_base2006 ratio individually, we know i7 ratio>i3 ratio>i5 ratio, which is match to question (b) and (c) performance result. Moreover, as we calculate ratio of SPECint_base2006 ratio, we can get similar result with (b) and (c) questions.

3. Assume a program requires the execution of 90×10^6 FP instructions, 110×10^6 INT instructions, 100×10^6 L/S instructions, and 25×10^6 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.

a). 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.

$$Original\ Cycle\ Count = (90 * 2 + 110 * 1 + 100 * 5 + 25 * 2) * 10^6 = 840 * 10^6$$

We need to accelerate program to run two time faster, which means New Cycle Count should cut down to $420*10^6$ cycles. However, FP instructions only affect at most $180*10^6$ cycles. It's impossible to let program run 2 times faster.

Use Amdahl's law to check this answer. $\frac{840*10^6}{4GHz} = 0.210s$. To run it 2 time faster, $0.105 = \frac{\frac{180*10^6}{4GHz}}{n} + \frac{660*10^6}{4GHz}$, we find 0.105 < 0.165 => n is impossible to be a positive number. We check it is impossible to run 2 time faster again.

b). 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.

$$New\ Cycle\ Count = (90 * 0.69 * 2 + 110 * 0.69 * 1 + 100 * 0.23 * 5 + 25 * 0.23 * 2) * 10^6 = 326.6 * 10^6$$

$$Improving\ exe.time = \frac{(840 - 326.6) * 10^6}{4GHz} = 128.35ms$$

4. 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 × volatge^2 × clock frequency

static power = voltage × leakage current

a). Find the average capacitive load.

$$Avg.Capacitive\ load = \frac{2 * 90}{1.25^2 * 5GHz} = 23.04nF$$

b). Find the percentage of the total dissipated power comprised by dynamic power and the ratio of static power to dynamic power.

$$Percentage = \frac{90}{90 + 60} * 100\% = 60\%$$

$$Ratio = \frac{60}{90} = \frac{2}{3}$$

c). If the total dissipated power is to be reduced by 35%, how much should the voltage be reduced to maintain the same leakage current?

$$New\ Power = 150 * 0.65 = 97.5$$

$$New\ Power = dynamic\ power + static\ power = \frac{1}{2} * 23.04 * 10^{-9} * v^2 * 5GHz + v * \frac{60}{1.25}$$

$$=> v = \frac{5\left(\sqrt{43} - 2\right)}{24} \approx 0.949\ V$$

Reduced voltage = 1.25-0.949 = 0.301V

5. Assume that a 25 cm diameter wafer has a cost of 15, contains 120 dies, and has 0.02 defects/cm^2.

Round to 4th decimal place

a). Find the yield for this wafer using the equation on page 50 of the Lecture 1-1.

$$Yield = \frac{1}{\left(1 + \left(\frac{0.02 * \left(\frac{25}{2}\right)^2 * \pi}{120 * 2}\right)\right)^2} = 0.9229$$

b). Find the cost per die for this wafer.

$$Cost = \frac{15}{120 * Yield} = 0.1354$$

c). 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.

$$Die\ Area = \frac{\left(\frac{25}{2}\right)^2 * \pi}{120 * 1.2} = 3.4088 cm^2$$

$$Yield = \frac{1}{\left(1 + \left(0.02 * 1.35 * \frac{3.4088}{2}\right)\right)^2} = 0.9139$$