Prof. Dr. Armin Biere
Dr. Mathias Fleury

Freiburg, 28/10/2020

Computer Architecture
Exercise Sheet 1 (v9)

Exercise 1

Give at least six applications of computers where correct operation is absolutely necessary. Also name at least one field of applications in which correct operation is not so important.

Exercise 2

When a program is adapted to run on multiple processors in a multiprocessor system, the execution time on each processor is comprised of computing time and the overhead time required for locked critical sections and/or to send data from one processor to another.

Assume a program requires $t = 100s$ of execution time on one processor. When run on p processors, each processor requires $\frac{t}{p}s$ and for each processor an additional $4s$ of overhead is required to receive the information, irrespective of the number of processors. Compute the per-processor execution time for 2, 4, 8, 16, 32, 64, and 128 processors. For each case, list the corresponding speedup relative to a single processor. Also calculate the ratio between the ideal speedup (speedup if there was no overhead) versus actual speedup. Which is the ideal number of processors?

(Taken from "Computer Organization and Design" by David A. Patterson, John L. Hennessy)

Exercise 3

Your are working on servers and you are updating CPUs. Your servers are running two kind of jobs: cryptography and HTTP-answering. While looking at the newest CPUs, you discover that they support new instructions but are not faster. The more advanced the CPU, the more instructions it supports, making your cryptography problem faster.

Your program has 80 instructions that can make use of the very new instructions while 20 cannot. One SIMD-1 instruction corresponds to 8 instructions in the old instructions, while one SIMD-2 instruction corresponds to 4 SIMD-1 instructions.[1] While porting you realize that you need 5 additional normal instructions.

[1] SIMD instructions work on multiple registers. This is not parallelism because only one core is involved. More details in the lecture.

	Normal	SIMD-1	SIMD-2
CPI for class	1	2	3
Prog 1	20+80		
Prog 2	25+0	10	
Prog 3	25+0	1	3

The CPU speed in 2 GHz, except if any SIMD-2 is run. If so, it becomes 1.5 GHz for all the instructions run at the same time.

a) Can you explain why the programs need new normal instructions?

b) Calculate the average CPI for all three programs.

c) Calculate the CPU time for all three programs.

d) We assume that the average speed is now 1.75 GHz. Calculate the CPU time for the last program.

(This exercise was loosely inspired by the next exercise, with the key difference that the frequency is not fixed there!)

Exercise 4

For simplicity, we assume that CPI is one.

You are running a CPU with 4 cores running at 4 GHz. You are running two kind of processes: (i) some cryptography taking 2 billion instructions (one core, cannot be parallelized) and (ii) answering some web requests also taking 20 billion instructions (can be perfectly parallelized).

You decide to update it to the newest CPU running at 4 GHz (no speed difference). This provides one new kind of instructions, making your cryptography program using only half of the instructions of the original program, i.e., 1 billion instructions. The webrequest program is not changed.

However there is a catch: If you run those kind of instructions, the clock speed of the entire chip goes down to 2 GHz to avoid overheating/consuming too much power.[2] Is it worth using the new instructions?

Same question with 8 billion instructions (8 Ginstruction) for the cryptography process.

(inspired by `https://blog.cloudflare.com/on-the-dangers-of-intels-frequency-scaling/` *: Intel's AVX2 instructions costed Cloudflare around 2 cores per 48-core CPU according to their measurement.)*

Exercise 5

We are interested in finding a faster matrix multiplication scheme (or prove that none exists) for matrices of size 3. To do so we encode our problem to SAT and solve it using the SAT solver Kissat. The encoding depends on a parameter p stating (roughly) how many operations are allowed. If you allow too many operations, then a very unefficient scheme can be derived. If too few operations

[2] In the lecture we only talked about instructions taking several cycles, not instructions slowing down the CPU.

are allowed, no scheme exists to multiply two matrices. If $p = 24$, we can find a scheme that is equivalent to the usual matrix multiplication. If there is a scheme for $p < 24$, there is a more efficient way to multiply matrices (or you can prove that none exists).

Kissat spends 80% of the time doing propagation. For theoretical reasons, this cannot be parallelized, while all other things can be.

Solving the problem for $p = 10$ takes 20 s and increasing p by one makes the problem 20 times harder.

a) We are interested in $p = 23$. Give an order of magnitude (in power of 10) of the amount of time required to solve the problem. Do we have any chance to solve the problem?

b) We parallelize everything we can. Calculate the runtime for 1, 2, 4, 8, and 40 processors for $p = 10$.

c) What is the fastest that we can make Kissat assuming everything is fully parallelized? Is the problem now solvable?

d) Actually most propagations that are done are not used subsequently. Assume that we can get rid of 15/16 of them. Are we closer to solving the problem?

e) What layers needs to be improved in order to solve the problem (algorithm/programming language/compiler/processor/instruction set)?

No submission is needed. The exercise sheet will be discussed on October 27th, 2022 in the online exercise session, 10:00 at:
`https://uni-freiburg.zoom.us/j/65775356475?pwd=dmUvei8ybDN4RFlmT1JUZnRtYlBGZz09`