Exercises

1. For the following four statements, state if the statement is correct or incorrect. If you think it is incorrect, explain why and what a correct statement would be.

   (a) Around 2006, *Moore's* law did not hold anymore. As a consequence, larger processor manufacturers started to produce multicore processors.

   (b) The term *power wall* typically means that the maximum clock speed has been reached and that chips would be too hot if they were clocked at higher frequencies.

   (c) Programming concurrent programs that achieve high parallel execution is a hard, but recently solved problem.

   (d) Before the multicore era, speedup by executing programs in parallel was basically not possible.

   Your solution:

A) F, moores lag är fortfarande sant tillsvidare; ett tag till.

B) T

C) F, inte lost ännu.

d) F, Det fanns fortfarande multiprocessorer som kunde göra.

2. Assume that you have a program that would take 20s to execute if you were running the program on infinitely many cores (if that was possible). Assume further that if you ran the program on 8 cores, you would get a speedup of 4. Assume also that N number of cores result in N times improvement for the part of the program that is possible to parallelize. What would then the speedup be if you executed the program on 20 cores?

   Your solution:

$$4 = \frac{TA+20}{20+\frac{TA}{8}} \Rightarrow TA+20 = 80+\frac{TA}{2} \Rightarrow TA = 120$$

$$20 \text{ cores} \Rightarrow \frac{120+20}{20+\frac{120}{20}} = \frac{140}{26} = \frac{70}{13} = 5,38$$

3. Consider the MIPS code below:

```
1    addi $t0,$t1,1
2    sub  $t1,$s0,$s1
3    add  $s1,$t0,$t1
4    xor  $t0,$t1,$s1
```

   (a) Assume that we execute the code on a superscalar MIPS-processor with four functional units. What is then the maximal *instructions per clock cycle (IPC)* for the code snippet? Note that the IPC is computed as the average instructions per clock cycles when executing the above four instructions.

   (b) What kind of dependency must an out-of-order processor take care of if it switches the order of the `addi` and the `sub` instructions? How can the processor resolve the hazard that occurs if the instructions are swapped?

Your solution:

A) $\frac{4}{3}$ instruktioner per klockcykel

B) Out-of-order processorer för att lösa hazards genom att köa i förväg för att hitta instruktioner med konfliktregister

4. Categorize the following keywords and concepts into one of the following two scenarios. Choose the scenario that fits best.

   Keywords and concepts: SIMD, MIMD, hardware multithreading, VLIW, cache coherence, Advanced Vector Extension (AVX), MapReduce, and dynamic multiple issue.

   - Scenario #1: A computer with a shared memory superscalar multicore (8 cores) processor with simultanious multithreading (SMT). The processor has separated L1 caches and a common L2 cache. The computer uses a modern Linux operating system.
   - Scenario #2: A cluster system with 1000 computers connected by an Ethernet network. Each computer has an uniprocessor with static multiple issue and data-level parallelism at the instruction level. The processors have separated instruction and data caches. The cluster has software for massive big data computations.

Your solution:

#1, Hardware multithreading, MIMD, cache coherence, dynamic multiple issue

#2) VLIW, AVX, MapReduce, SIMD

5. Explain the differences between a *process* and a *thread* for a modern operating system. The answer should include and clarify the following keywords: virtual memory, memory protection, files, global data, stack, and multicore.

Your solution:

Skillnaden mellan de är att en process har ett eget virtuelt minne vilket gör att det har ett memory protection från de andra processerna i det fysiska minnet. processer använder dessutom olika trådar för att köra program, med hjälp av multicore kan man ha flera processorenheter på samma chip. En annan skillnad är att proceser kan hantera och använda global data så som funktioner och filer. stack används som en vanlig datastruktur som många program använder.

6. Assume that you are writing a multithreaded program in an imperative C-like language. You have one function `foo()` that takes zero arguments and does not return a value, but it has a side effect. This function is called from several threads. Explain and give an example with pseudo code for how you can make this function thread safe.

Your solution:

```
Void foo();
{
  öppna ngt;
exempel_vänta(lås);
importera saker;
exempel_efter(lås);
}
```

exempel används för att låsa en funktion som exekveras av en tråd som försöker anropas.

Efter funktionen återställs exempler.
Exemplet är ett så länge funktionen är ledig och noll när funktionen har körts

Corrected by _____. Total number of points: _____