Exercises 6
Parallel Processors and Programs

Computer Organization and Components / Datorteknik och komponenter (IS1500), 9 hp
Computer Hardware Engineering / Datorteknik, grundkurs (IS1200), 7.5 hp

KTH Royal Institute of Technology
Friday 18th December, 2020

Concurrency, Parallelism, and Concepts

1. Consider the following three cases:

   (a) An embedded system consisting of a MIPS microcontroller (uniprocessor) that runs two periodic tasks with the period times 5ms and 20ms, respectively. The tasks are invoked using a timer interrupt.

   (b) A SIMD uniprocessor that solves numerical problem and achieves high speedup because several floating-point instructions can be executed in parallel.

   (c) A web server that is implemented on an Intel Core i7 processor system with 4 cores. Each time a HTTP request comes to the server, the web server creates a new thread that handles the request.

   For each of the three systems, state if this system executes programs sequentially or concurrently and if the hardware is executing in parallel or not. Motivate your answer.

2. Draw a 2 by 2 matrix, where each element represents SISD, SIMD, MISD, and MIMD.

   (a) Explain what the acronyms stand for.

   (b) Place the following words and acronyms inside the matrix and explain why they belong in one or more places: MIPS uniprocessor, task-level parallelism, AVX, Intel Core i7, GPU, data-level parallelism, and ILP.

Speedup and Amdahl's law

3. Assume that you are performing a speedup performance measurement for a image processing algorithm. Someone else has created a very efficient sequential implementation I_s and you have implemented a new parallel version I_p of the algorithm. The sequential implementation has been executed on a benchmark B on a computer C_1 with two cores, each running at 4GHz. You have the source code for both I_s and I_p and access to a machine with 8 cores. Each core is running at 2.2GHz with hardware multi-threading, which makes the OS believe that there are 16 cores in the machine.

   Explain how you would perform a fair speedup evaluation of your parallel implementation.

4. Assume we have a program where 10% of the execution time is purely sequential and that the rest of the execution time can be improved by parallelization. For the part of the code that can be parallelized, each core gives only 80% improvement. For instance, 5 cores give $5 \times 80\% = 4$ times improvement.

   (a) Create a speedup chart, showing speedup on the Y-axis and the number of cores on the X-axis. Show the graph for 1 to 200 cores, for instance by plotting with 25 cores interval.

   (b) What is the maximal speedup that can be achieved regardless how many cores we add?

   (c) What is it called if we would increase the problem size linearly to the number of cores? What kind of scaling was used in problem (a)? Why would either of these scaling approaches make sense?

Instruction Level Parallelism

5. Consider the following C function.

```
void reverse_double(int *src, int *dst, int n){
  int i;
  for(i=0; i < n; i++){
    dst[i] = src[n - i - 1] * 2;
  }
}
```

   (a) Explain what the C function is doing.

   (b) Translate the C program into MIPS code, which is executed on a 1-issue MIPS processor. Assume the conditional branch calculation for `beq` can be computed in the decode stage.

   (c) Compute the IPC for executing the function with $n = 5$. Do not include the cost of calling the function, but include the return cost. Assume an 1-issue 5 stage pipelined MIPS processor with static branch prediction assuming branch-not-taken. The comparison for `beq` is assumed to be done in the decode stage.

   (d) Assume we have a static 2-issue processor that can execute any type of instruction in two different slots. Show the optimized MIPS code for each slot.

   (e) Compute the IPC again for executing the function on the 2-issue processor with $n = 5$. Again, do not include the cost of calling the function, but include the return cost.

   (f) What speedup do we achieve by using a 2-issue processor in this case?

Concurrent Programming and Semaphores

6. In this task, you should consider a multi-threaded producer-consumer problem. There are two threads, a producer thread and a consumer thread. The producer thread is writing data into a first-in-first-out (FIFO) buffer and the consumer thread is reading from the buffer. The FIFO buffer can hold between 0 and n elements.

   The task is to create both a tread safe consumer function and a thread safe producer function with proper synchronization. If the buffer is empty, the consumer needs to wait until there is an available item in the buffer. If the buffer is full (holds n elements), the producer has to wait until there is space available, before it writes any data items into the buffer. Solve the problem by using semaphores. You may write the solution as pseudocode, as long as you clearly explain the program semantics.

Data-Level Parallelism

7. Consider the following lines of assembly code:

   ```
   vmovapd  (%r10), %ymm0
   vmovapd  (%r11), %ymm1
   vmulpd   %ymm0,  %ymm1, %ymm1
   vaddpd   (%r10), %ymm1, %ymm1
   vmovapd  %ymm1, (%r11)
   ```

   (a) What kind of assembly code is this?

   (b) Explain what each line of code is doing.