**1541 Homework 3 Parallel Processing**

**Name: Andrew Tran\_\_\_\_\_\_\_\_\_\_\_**

**Due: Apr. 17, 2019**

1. Let A = a matrix of *m×n*,B = a matrix of *n×p*, C = A×B. Assume we are going to compute C on both a single core shared memory machine and a 4-core shared-memory machine. Express the speedup we would expect to obtain on the 4-core machine in *m, n, p*, ignoring any memory issues.

* There are mp cells in the product matrix
* Each cell requires 2n operations (n mults, n adds)
* If each core is assigned a different cell to work on, we would expect to get a 4 times speedup in a 4-core machine compared to a single core machine.

Single Core: mpn + mpn = 2mpn

4 Cores: (2mpn/4) = (1/2)mpn

Speedup = Single Core/4 cores = **4**

1. Consider the following portions of parallel programs running at the same time on four processors in a symmetric multicore processor. Assume that before this code is run, both x and y are 0.

Core 1: x=2;

Core 2: y=2;

Core 3: w=x+y+1;

Core 4: z=x+y;

* + 1. What are the all possible resulting values of w, x, y, and z? For each possible outcome, explain how we might arrive at those values.

Answer: problem 6.7 in the textbook

|  |  |
| --- | --- |
| Value | Reason |
| X = 2 | X and Y have no dependencies, so their final values are deterministic |
| Y = 2 |
| W = 1 | If the thread in core 3 finishes before cores 1 and 2 |
| W = 3 | If either core 1 or core 2 finishes before core 3 reads x or y |
| W = 5 | If both cores 1 and 2 finish before core 3 reads x and y |
| Z = 0 | If the thread in core 4 finishes before cores 1 and 2 |
| Z = 2 | If either core 1 or core 2 finishes before core 4 reads x or y |
| Z = 4 | If both cores 1 and 2 finish before core 4 reads x and y |

* + 1. Explain how could you make the execution more deterministic so that only one set of values is possible?

Synchronize cores 1 and 2 before running cores 3 and 4. This will ensure that the dependencies in cores 3 and 4 are determined before they run.

1. Consider the following three CPU organizations:

CPU SS: A 2-core superscalar microprocessor that provides out-of-order issue capabilities on 2 function units (FUs). That is, each core can concurrently run up to two independent operations each cycle. Only a single thread can run on each core at a time.

CPU MT: A fine-grained multithreaded processor that allows instructions from 2 threads to be run concurrently (i.e., there are two functional units), though only instructions from a single thread can be issued on any cycle.

CPU SMT: An SMT processor that allows instructions from 2 threads to be run concurrently (i.e., there are two functional units), and instructions from either or both threads can be issued to run on any cycle.

Assume we have two threads X and Y to run on these CPUs that include the following operations:

|  |  |
| --- | --- |
| Thread X | Thread Y |
| A1 – takes 3 cycles to execute | B1 – take 2 cycles to execute |
| A2 – no dependences | B2 – conflicts for a functional unit with B1 |
| A3 – conflicts for a functional unit with A1 | B3 – depends on the result of B2 |
| A4 – depends on the result of A3 | B4 – no dependences and takes 2 cycles to execute |

* + 1. Assume that you have 1 SS CPU. How many cycles will it take to execute these two threads?

Assuming each instruction takes 1 cycle to complete unless stated otherwise

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Core 1 (Thread X) | | Core 2 (Thread Y) | |
| Cycle # | FU 1 | FU 2 | FU 1 | FU 2 |
| 1 | A2 | A3 | B2 | B4 |
| 2 | A4 | A1 | B3 | (B4 done) |
| 3 | --- |  | B1 | --- |
| 4 | --- | (A1 done) | (B1 done) | --- |

**4 Cycles**

* + 1. Now assume you have 1 MT CPU. How many cycles will it take to execute these two threads?

|  |  |  |
| --- | --- | --- |
| Cycle # | FU 1 | FU 2 |
| 1 | --- | B1 |
| 2 | A1 | (B1 done) |
| 3 |  | B2 |
| 4 | (A1 done) | A2 |
| 5 | A3 | A4 |
| 6 | B3 | B4 |
| 7 | --- | (B4 done) |

**7 Cycles**

* + 1. Now assume now you have 1 SMT CPU. How many cycles will it take to execute these two threads?

|  |  |  |
| --- | --- | --- |
| Cycle # | FU 1 | FU 2 |
| 1 | A1 | B1 |
| 2 |  | (B1 done) |
| 3 | (A1 done) | B2 |
| 4 | A2 | B3 |
| 5 | A3 | B4 |
| 6 | A4 | (B4 done) |

**6 Cycles**