Matthew Strayhorn

CS1541

HW #6

Questions 1: Consider the parallel sum algorithm shown on page 29 of the class slides. Derive expressions for the speedup when the algorithm is applied to sum N numbers on P processors and:

(a) P is a power of 2 and N is a multiple of P.

(b) The only restriction is that N is larger than or equal to P. Plot in a graph the speedup that is expected when the algorithm is executed on 16 processors for N = 100, 200, 400, 800, 1000. Plot on the same graph the speedup curve when the number of processors is 64 rather than 16.

A. Ts(n) = n-1

Tp(n) = n/(p-1) + log2 p

Sp(n) = (n-1)/((n/(p-1)) + log2 p)

B. Ts(n) = n-1

Tp(n) = ((n/p)-1) + log2 p

Sp(n) = (n-1) / (((n/p)-1) + log2 p)

Question 2: Do problem 6.9 from the text book (problem 7.12 in the old version) – You should consider the case of fine grained MT as well as SMT when you consider 1MT CPU. Assume that instructions A1, A2, A4, B1, B3 and B4 can execute on any FU, but that A1 and A3 should execute on the same unit and B1 and B2 should execute on the same. Present your answer in the form of a time-line. For example, the execution of thread X on SS can be represented as: Cycle1 Cycle2 Cycle3 Cycle4 Cycle5 FU 1: A1 A1 A1 A3 A4 FU 2: A2 - - - - Note: The definition of CPU SS in the question is confusing. Use the following definition when solving the question: An SS CPU is a CPU with 2 functional units that provides out-of order issue capabilities.

6.9.1.

|  |  |  |
| --- | --- | --- |
|  | Core 1 | Core 2 |
| Cycle 1 | A1, A2 | B1, B4 |
| Cycle 2 | A1 | B1, B4 |
| Cycle 3 | A1 | B2 |
| Cycle 4 | A3 | B3 |
| Cycle 5 | A4 |  |

It takes 5 cycles to execute all of the instructions and 8 out of 20 issue slots are wasted

6.9.2.

Since there are only 2 threads, the second SS CPU will be idle and the other one will execute the instructions the same as above. It still takes 5 cycles to execute the instructions and 28 issue slots are wasted.

6.9.3.

|  |  |  |
| --- | --- | --- |
|  | Functional Unit 1 | Functional Unit 2 |
| Cycle 1 | A1 | B1 |
| Cycle 2 | A1 | B1 |
| Cycle 3 | A1 | B2 |
| Cycle 4 | A2 | B3 |
| Cycle 5 | A3 | B4 |
| Cycle 6 | A4 | B4 |

The MT CPU takes 6 cycles to execute these instructions and doesn’t waste any issue slots.

Questions 3: Compile and execute the “compute pi” program (in file pi.c -- see notes below) with 100000 sample points and P =1, 2, 4, 8, 16 and 32 threads on “ra.cs.pitt.pitt.edu”. Report the execution times in a table and draw the speed up curve. Repeat the experiment with 10000 and 1000000 points and compare the speed ups. Comment on your results.

|  |  |  |  |
| --- | --- | --- | --- |
| # of points | # of threads | Execution time | Speedup |
| 100,000 | 1 | 0.007762 | 0 |
| 100,000 | 2 | 0.004097 | 1.894557 |
| 100,000 | 4 | 0.002393 | 3.243627 |
| 100,000 | 8 | 0.001806 | 4.297896 |
| 100,000 | 16 | 0.00146 | 5.316438 |
| 100,000 | 32 | 0.00297 | 2.613468 |
|  |  |  |  |
|  |  |  |  |
| 10,000 | 1 | 0.001248 | 0 |
| 10,000 | 2 | 0.00096 | 1.3 |
| 10,000 | 4 | 0.000896 | 1.392857 |
| 10,000 | 8 | 0.000994 | 1.255533 |
| 10,000 | 16 | 0.00248 | 0.503226 |
| 10,000 | 32 | 0.003177 | 0.392823 |
|  |  |  |  |
|  |  |  |  |
| 1,000,000 | 1 | 0.071614 | 0 |
| 1,000,000 | 2 | 0.036024 | 1.987952 |
| 1,000,000 | 4 | 0.01843 | 3.88573 |
| 1,000,000 | 8 | 0.009745 | 7.348794 |
| 1,000,000 | 16 | 0.005401 | 13.2594 |
| 1,000,000 | 32 | 0.004423 | 16.19127 |

For both 10,000 and 100,000 sample point graphs, the speedups exhibited similar behavior where they would increase a lot initially and then peak and start decreasing. However for the 1,000,000 points graph, there was no decreasing behavior and it only increased as the number of threads increased.