Cognome Nome

_____

Advanced and parallel architectures

Prof. A. Massini

June 11, 2015

Exercise 1a (2 points)	
Exercise 1b (2 points)	
Exercise 2 (5 points)	
Exercise 3 (3 points)	
Exercise 4a (3 points)	
Exercise 4a (2 points)	
Exercise 5 (4 points)	
Exercise 6 (5 points)	
Question 1 (3 points)	
Question 2 (3 points)	
Total (32 points)	

Exercise 1 (2+2 points) – Loop Dependences

a) Explain what are output dependences and antidependences in a loop.

b) In the following loop, find all the output dependences and antidependences. Eliminate the output dependences and antidependences by renaming.

```
for (i=0;i<100;i++) {
X[i] = X[i] / Z[i];       /* S1 */
Z[i] = X[i] - k;          /* S2 */
X[i] = W[i] + k;          /* S3 */
W[i] = Y[i] / X[i];       /* S4 */
```

Exercise 2 (5 points) - Data Flow Machine

Consider the computation of the determinant *det* of a matrix $A=(a_{i,j})$ of size 3x3 on a Dataflow Machine.

Write the instructions needed to obtain *det* according to the Dataflow Machine format, group the instructions according to the parallel steps, and draw the diagram for the execution.

Exercise 3 (3 points) - GPU & CUDA

A CUDA device's SM (Streaming Multiprocessor) can take up to 2048 threads and up to 8 thread blocks. Which of the following block configuration would result in the most number of threads in the SM?
(A) 128 threads per block
(B) 256 threads per block
(C) 512 threads per block
(D) 1024 threads per block

Give a comment for each answer.

Exercise 4 (3+2 points) - GPU & CUDA

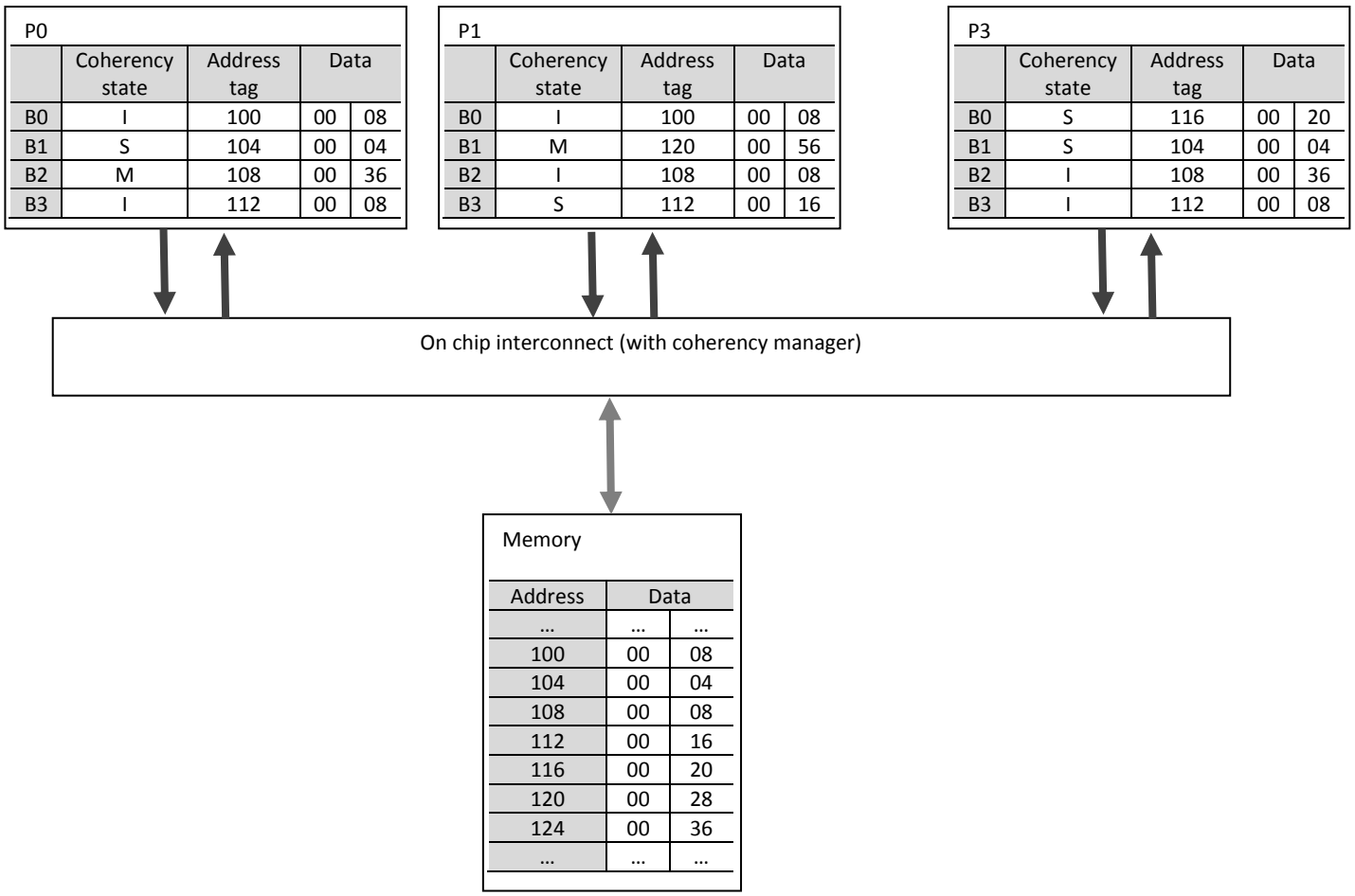
You need to write a kernel that operates on a matrix of size 500x800. You would like to assign one thread to each matrix element. You would like your thread blocks to use the maximum number of threads per block possible on your device, having compute capability 1.3, assuming bi-dimensional blocks.

a) How would you select the grid dimensions and block dimensions of your kernel?
b) How many idle threads do you expect to have?

Technical specifications	Compute capability (version)									
	1.0	**1.1**	**1.2**	**1.3**	**2.x**	**3.0**	**3.5**	**3.7**	**5.0**	**5.2**
Maximum dimensionality of grid of thread blocks	2				3					
Maximum x-dimension of a grid of thread blocks	65535					2^{31}-1				
Maximum y-, or z-dimension of a grid of thread blocks	65535									
Maximum dimensionality of thread block	3									
Maximum x- or y-dimension of a block	512				1024					
Maximum z-dimension of a block	64									
Maximum number of threads per block	512				1024					
Warp size	32									
Maximum number of resident blocks per multiprocessor	8					16			32	
Maximum number of resident warps per multiprocessor	24		32		48	64				
Maximum number of resident threads per multiprocessor	768		1024		1536	2048				
Technical specifications	**1.0**	**1.1**	**1.2**	**1.3**	**2.x**	**3.0**	**3.5**	**3.7**	**5.0**	**5.2**
	Compute capability (version)									

Exercise 5 (4 points) - Snooping protocol for cache coherence

Consider a multicore multiprocessor implemented as a symmetric shared-memory architecture, as illustrated in the figure.



P0

	Coherency state	Address tag	Data	
B0	I	100	00	08
B1	S	104	00	04
B2	M	108	00	36
B3	I	112	00	08

P1

	Coherency state	Address tag	Data	
B0	I	100	00	08
B1	M	120	00	56
B2	I	108	00	08
B3	S	112	00	16

P3

	Coherency state	Address tag	Data	
B0	S	116	00	20
B1	S	104	00	04
B2	I	108	00	36
B3	I	112	00	08

Memory

Address	Data	
…	…	…
100	00	08
104	00	04
108	00	08
112	00	16
116	00	20
120	00	28
124	00	36
…	…	…

Each processor has a single, private cache with coherence maintained using the snooping coherence protocol. Each cache is direct-mapped, with four blocks each holding two words. The coherence states are denoted M, S, and I (Modified, Shared, and Invalid).

Each part of this exercise specifies a sequence of one or more CPU operations of the form:
P#: <op> <address> [<value>]
where P# designates the CPU (e.g., P0), <op> is the CPU operation (e.g., read or write), <address> denotes the memory address, and <value> indicates the new word to be assigned on a write operation.

For each part of this exercise, assume the initial cache and memory state as illustrated in the fgure and treat each action as independently applied to the initial state shown in the figure.
Show in the table for the results:
- miss/hit
- the coherence state before the action
- the CPU processor Pi and cache block Bj
- the changed state (i.e., coherence state, tags, and data) of the caches and memory after the given action.
Specify the value returned by a read operation.

a) P0: write 104 ← 24

hit/miss	state before	Pi.Bj (state, tag, datawords)

b) P1: read 108

hit/miss	state before	Pi.Bj (state, tag, datawords)

value returned by read:

c) P0: write 124 ← 12

hit/miss	state before	Pi.Bj (state, tag, datawords)

d) P0: write 116 ← 32

hit/miss	state before	Pi.Bj (state, tag, datawords)

Comments

Exercises 6 (5 points) Performance equation & Amdhal Law

Consider a machine having a clock rate of 400 MHz. The following measurements are recorded with respect to the different instruction classes for the instruction set running a given set of benchmark programs:

Instruction Type	Instruction Count (millions)	Cycles per Instruction
Arithmetic and logic	6	1
Load and store	4	3
Branch	1	4
Others	5	3

a) Determine the effective CPI and execution time. *(2 points)*

b) Knowing that we can express the MIPS rate in terms of the clock rate and CPI as follows:

$$\text{MIPS rate} = \frac{IC}{T \times 10^6} = \frac{f}{CPI \times 10^6}$$

determine the MIPS rate for the machine above. *(1 points)*

c) Assume that instructions load and store can be modified so that they take 1 cycle per instruction instead of 3 as in the table. Compute the speed up obtained by introducing this enhancement using the Amdhal law. *(2 points)*

Question 1 (3points)

Describe the Looping algorithm for the Benes network

Question 2 (3 points)

Describe the directory based protocol for Cache coherence