Advanced and parallel architectures

Prof. A. Massini

Exam – June 26, 2015

Cognome Nome

_____

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

Part A

Exercise 1a (2 points) – Number representation
Given the values A= 00 10 00 11 00 10 and B= 11 10 10 01 11 01 in the signed RB representation, convert A and b in decimal.

Exercise 1b (4 points) – Number representation
Describe the procedure to verify if a value in the RB (Redundant Binary) representation is equal to 0.
Show step by step how to apply the procedure to verify if A+B (Exercise 1a) is equal to zero.

Exercise 2 (3 points) – Instruction pipeline

Consider the following loop expressed in a high level language:

```
for (i =0; i < N; i ++) {
        vectA[i] = vectB[i]
        vectB[i] = vectB[i] + K1 + K2;
        vectC[i] = vectB[i]
}
```

The program has been compiled in MIPS assembly code assuming that registers $t6 and $t7 have been initialized with values 0 and 4N respectively.

The symbols VECTA, VECTB and VECTC are 16-bit constant.

Let us consider the loop executed by 5-stage pipelined MIPS processor without any optimisation in the pipeline.

1. Identify the Hazard Type (Data Hazard or Control Hazard) in the last column
2. In the first column identify the number of stalls to be inserted before each instruction (or between stages IF and ID of each instruction) necessary to solve the hazards
3. For each hazard, add an ARROW to indicate the pipeline stages involved in the hazard

Num. Stalls	INSTRUCTION	C1	C2	C3	C4	C5	C7	C6	C8	C9	C10	C11	C12	C13	Hazard Type
	`beq $t6,$t7,END`	IF	ID	EX	ME	WB									
	`lw $t2,VECTB($t6)`		IF	ID	EX	ME	WB								
	`sw $t2,VECTA($t6)`			IF	ID	EX	ME	WB							
	`addi $t2,$t2,K1`				IF	ID	EX	ME	WB						
	`addi $t2,$t2,K2`					IF	ID	EX	ME	WB					
	`sw $t2,VECTB($t6)`						IF	ID	EX	ME	WB				
	`sw $t2,VECTC($t6)`							IF	ID	EX	ME	WB			
	`addi $t6,$t6,4`								IF	ID	EX	ME	WB		
	`j FOR`									IF	ID	EX	ME	WB	

Exercise 3 (4 points) – Pipelined operations

Show the scheme and the execution of the pipelined multiplications: 3x4 and (-2)x3. Verify the results.

Exercise 4a (3 points) – Interconnection networks

Complete the scheme of the Butterfly and Baseline and explain how works the self-routing. Show the switch setting to realize permutation $P = \begin{pmatrix} 01234567 \\ 72610435 \end{pmatrix}$

Exercise 4b (3 points) – Interconnection networks

Complete the scheme of the Baseline-Baseline^{-1} and on a Butterfly-Butterfly^{-1}. Show the switch setting to realize permutation P on a Baseline-Baseline^{-1} and on a Butterfly-Butterfly^{-1} and explain how it is obtained.

Exercise 4c (3 points) – Interconnection networks

Draw a Hypercube of dimension 4 and show (on different pictures) the possible routings between nodes 1110 and 0010.
Explain how the paths are obtained.

Question 1 (5 points)

Explain what is an interconnection network and its feature (such as node degree, diameter, scalability, routing algorithm).

Question 2 (5 points)

Describe the Modified Number System MSD.

Advanced and parallel architectures

Prof. A. Massini

Exam – June 26, 2015

Cognome Nome

_____

Exercise 1a (2 points)	
Exercise 1b (3 points)	
Exercise 2a (2 points)	
Exercise 2b (2 points)	
Exercise 3 (4 points)	
Exercise 4 (4 points)	
Exercise 5a (2 points)	
Exercise 5b (4 points)	
Question 1 (5 points)	
Question 2 (5 points)	
Total (33 points)	

Part B

Exercise 1a (2 points) – GPU

Given a matrix A of size NxN, the following code compute the transpose Y of A by using the 1D indexing instead of the 2D one.

```
void transpose(float in[][], float out[][], int N)
{
for (int j=0; j < N; j++)
       for(int i=0; i < N; i++)
       Y[i*N+j] = A[j*N+i];
       }
```

Parallelize over rows, 1 thread per row and replace outer loop with index calculation (write the expression of i in the following code)

```
global__ void gpuTranspose_kernel(int rows, int cols, float *in, float *out)
{
int i, j;
      i =
      for (int j=0; j < N; j++)
      out [ i * rows + j ] = in [ j * cols + i ];
}
```

Explain.

Exercise 1b (3 points) – GPU

Consider a matrix 1500x1500. You would like to assign one thread to each matrix element. You would like your thread blocks to be square. How would you select the grid dimensions and block dimensions of your kernel to minimize the number of idle threads on a device having compute capability 3.0?

And on a device having compute capability 1.3?

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 2a (2 points) – Loop dependences

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

```
for (i=1; i<=n; i++)
{
Z[i]=X[i]+2;        \*S1*\
W[i]=Y[i]-1;        \*S2*\
X[i]=U[i]+V[i];     \*S3*\
X[i]=Z[i]+3;        \*S4*\
Y[i]=X[i]*2;        \*S5*\
}
```

Exercise 2b (2 points) – Loop dependences

In the following loops find the loop carried dependences with respect to index i and/or index j

```
for (i=1; i<n; i++) {
  a[i] = a[i-1] + 1;      \*S1*\
  b[i] = a[i];            \*S2*\
}

for (i=1; i<n; i++)
  for (j=1; j< n; j++)
    a[i][j] = a[i][j-1] + 1;   \*S3*\

for (i=1; i<n; i++)
  for (j=1; j< n; j++)
    S4: a[i][j] = a[i-1][j] + 1; \*S4*\
```

Exercise 3 (4 points) - Data Flow Machine

Consider the computation executed in the loop:

```
for (i =0; i < N; i ++) {
vectA[i] = vectB[i]
vectB[i] = vectB[i] + K1 + K2;
vectC[i] = vectB[i]
}
```

where vectA, vectB and vect C have size N=3.

Write the instructions needed on a Dataflow Machine to obtain the three vectors, having vector, K1 and K2 as input, and eliminating the for loop.

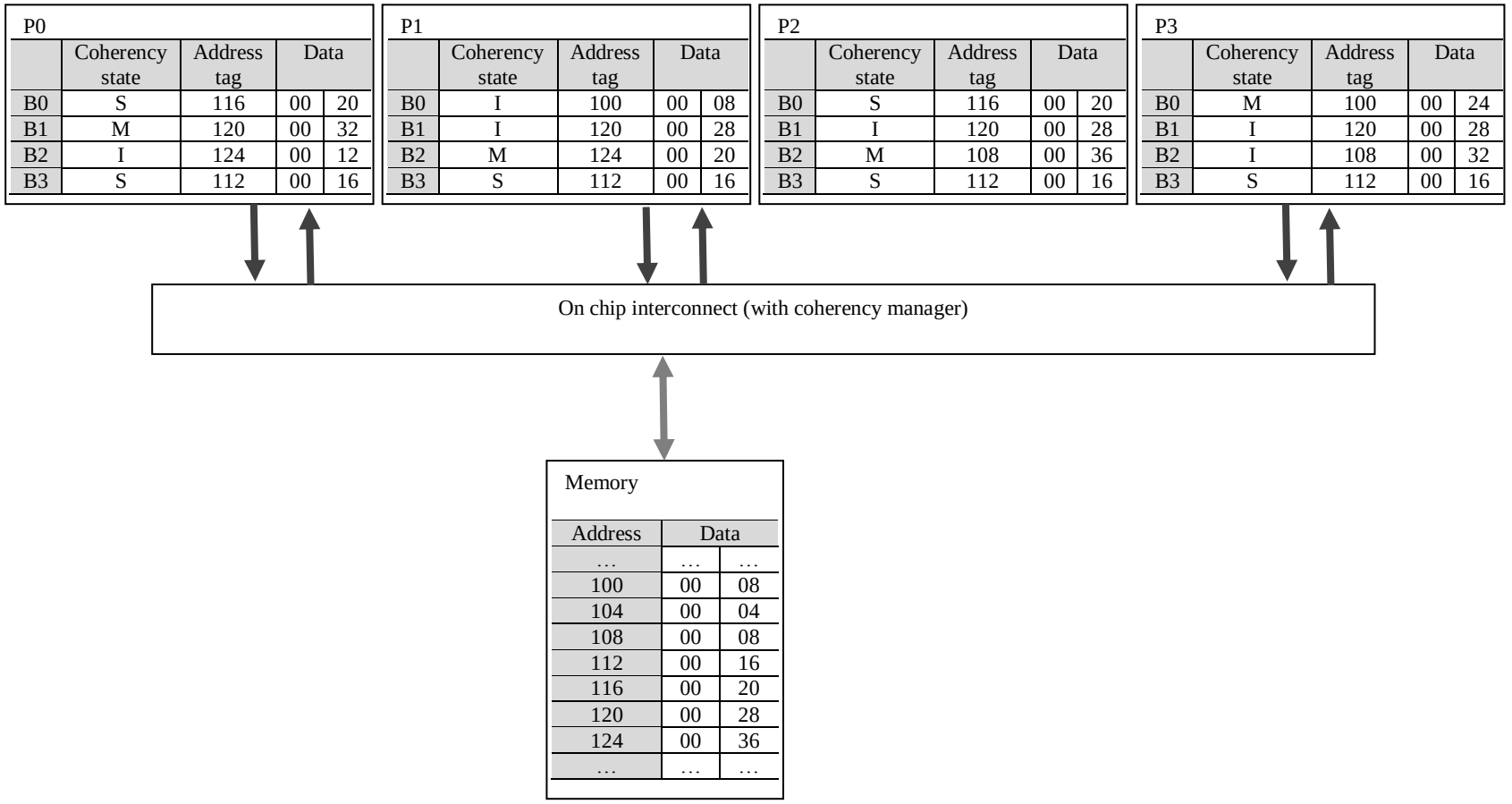
Group the instructions according to the parallel steps, and draw the diagram for the execution.

Exercise 4 (4 points) - Cache coherence

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

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.



P0

	Coherency state	Address tag	Data	
B0	S	116	00	20
B1	M	120	00	32
B2	I	124	00	12
B3	S	112	00	16

P1

	Coherency state	Address tag	Data	
B0	I	100	00	08
B1	I	120	00	28
B2	M	124	00	20
B3	S	112	00	16

P2

	Coherency state	Address tag	Data	
B0	S	116	00	20
B1	I	120	00	28
B2	M	108	00	36
B3	S	112	00	16

P3

	Coherency state	Address tag	Data	
B0	M	100	00	24
B1	I	120	00	28
B2	I	108	00	32
B3	S	112	00	16

Memory

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

For each part of this exercise, assume the initial cache and memory state as illustrated in the figure and **treat actions as applied one after another starting from 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) P1: write 120 ← 24

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

b) P0: write 104 ← 08

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

c) P3: read 108

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

d) P2: write 100 ← 20

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

Comments

Exercise 5a (2 points) - Performance

Consider 3 processors P1, P2 and P3 with clock rates and CPI given below, are running the same program:

	clock rate	**CPI**
P1	2 GHz	1.5
P2	1.5 GHz	1.0
P3	3 GHz	2.5

Which processor will have the best performance?

Exercise 5b (4 points) – Amdhal Law

Three enhancements with the following speedups are proposed for a new architecture:

$Speedup_1 = 30$

$Speedup_2 = 20$

$Speedup_3 = 10$

Assume for some benchmark, the fraction of use is 15% for each of enhancements 1 and 2 and 70% for enhancement 3. We want to maximize performance. If only one enhancement can be implemented, which should it be? If two enhancements can be implemented, which should be chosen?

Question 1 (5 points)

Explain the Amdhal Law and Performance Equation.

Question 2 (5 points)

Describe the Vector Architecture.