

CS 573
Final Exam
May 12, 2006

The following exam is open book and open notes. You may feel free to use whatever additional reference material you wish, but **no electronic aids** are allowed. Please note the following instructions. There will be a ten point deduction for failure to comply with them:

- start each problem on a new sheet of paper
- write your social security number, but not your name, on each sheet of paper you turn in
- show your work whenever appropriate. There can be no partial credit unless I see how answers were arrived
- be succinct. You may lose points for facts that, while true, are not relevant to the question at hand

You have until 10:00 to finish the exam. The questions are equally weighted.

1. One of the most striking features of the Cray-1 instruction set was its use of vector instructions, which would execute an operation on corresponding elements of arrays. So, for instance,

```
vmul V1, V2, V3
```

would effectively execute the loop

```
for (i = 0; i < 64; i++)  
    V1[i] = V2[i]*V3[i];
```

- (a) Why did these instructions give the Cray-1 a performance advantage?

The main thing they did was provide a way to keep the Cray's pipelines full. When one of these vector instructions was executed, once the startup latency was done with results could occur at a rate of one per cycle. Chaining the instructions gave a further speedup, since it meant that the startup latency could be avoided much of the time.

- (b) Would they still result in an advantage for a modern computer capable of out of order execution, such as a MIPS R10000 or a Pentium 4?

Not really. Modern processors are able to use their scoreboarding and branch prediction to keep the pipelines pretty much full without using vector instructions.

Huge numbers of people "answered" part (a) giving me a nice little essay on chaining. Well, chaining does provide further improvements beyond what you'd get from vector instructions without chaining, but these answers never said a word that actually addressed the question.

I gave the two parts of the question ten points each; a minimally responsive answer (so, something that at least had something to do with the Cray architecture as opposed to, say, their FORTRAN compiler) to one of the parts got 5.

2. In an Alewife system variable x is homed on processor P1, and initially has a processor set $P = \{1\}$. Suppose the following sequence of memory operations takes place:

P2:	$w(x)1$
P3:	$r(x)1$
	$w(x)2$

What are the messages that must be passed among the processors to accomplish this?

P3->RREQ->P1
P1->RDATA->P3

P2->WREQ->P1
P1->INV->P3
P3->ACKC->P1
P1->WDATA->P2

P3->WREQ->P1
P1->INV->P2
P2->UPDATE->P1
P1->WDATA->P3

Points	Error
-2	Missing or extra message
-1	Wrong message type
-5	Put all P2 interactions before P3 interactions

3. Suppose the following programs are executed on a multiprocessor using an MESI snoopy cache protocol, in the order shown (so P1 executes its program, then P2 executes its program). What will the states of the variables be in the two processors' caches after this? Assume the three variables map to different cache blocks, and initially the caches are all empty. I realize z is being used uninitialized; that's OK.

P1: $y = 1;$
 y is in P1 in state **M**.
P2: $x = y + z;$
 y is changed to state **S** in P1 and P2
 z is changed to state **E** in P2
 x is changed to state **M** in P2

To summarize, after we're done the states are:

	P1	P2
x	I	M
y	S	S
z	I	E

Points	Error
-5	Cache block in wrong state
-5	No actual summary of states, but easy to derive from description

4. Here are some terms that appeared in the wormhole routing paper. What do they mean?

(a) Minimal (in the sense of a minimal routing algorithm)

A minimal routing algorithm is one that will always produce the shortest possible path from source to destination.

(b) Deadlock

Deadlock is a situation in which several processors are competing for resources in such a way that they all have some of the resources then need, but need resources held by other processors which will not give them up.

In this context, it's a situation in which the routing algorithm selects routes for a set of messages such that they are blocking each other's progress.

Nearly everyone got these. The exceptions were someone who buried the definition of minimal in over a page of description of wormhole routing, and somebody who said deadlock was bad, but never got around to defining it.

5. Consider the following three one-line programs:

P1 P2 P3
 $x = 1; \quad y = 2; \quad z = x + y;$

- (a) Using the notation we've been using to show memory operations (as in Problem 2), show that it is possible, under lazy release consistency, for **P1** and **P2** to execute before **P3** but for z 's final result to be 2. Assume x , y , and z are all in separate cache blocks.

The hard part with this one isn't actually showing that z 's final value might be 2, it's showing that it won't be 0. The point is just that values need not be updated until an acquire/release pair takes place; however, there is nothing in the definition to keep some of the data from being sent around. So the following is perfectly legal:

P1: $W(x)1$		
P2: $W(y)2$		
P3:	$R(x)0$	$R(y)2 \quad W(z)2$

On reflection, since the implementations of LRC we've discussed have emphasized not sending the data until acquire-time rather than the definition of the model, if somebody claims z 's final value has to be 0 I'll accept it.

Probably the most common errors here are a combination of inserting acq/rel pairs (my code doesn't have them – and the result I'm asking for depends on a delay in propagating values) with then assuming values don't get propagated properly. No, if there is an acq/rel chain between the writes and the reads, the last values written will be read.

- (b) Now add appropriate barrier operations to ensure that z 's final value is 1. Show both the modified programs and the new memory operations diagram.

P1	P2	P3
$x = 1;$	$S;$	$S;$
$S;$	$S;$	$z = x + y;$
$S;$	$y = 2;$	$S;$

I'm using S to represent a barrier here. what happens in words is that $P2$ and $P3$ both get hung up on barriers while $P1$ executes $x=1$. $P1$ now reaches its barrier; the new value of x is propagated to $P3$. $P1$ and $P2$ now get hung up on the second barrier while $P3$ calculates z (using a value of 0 for y). Now $P3$ gets to its barrier, and all three processes continue.

Points	Error
-3	Added Acq/Rel in part (a)
-2	Added invalidate operations???
-3	Acq/Rel in part (a) didn't meet LRC spec
-5	Don't show what gets read and written
-5	Doesn't completely sync up variables
-8	No barriers added in (b)
-3	Only one barrier added in (b)

6. The following 13 bits contain eight bits of data, 4 check bits, and one overall parity bit (so it's a single-error correcting, double-error detecting code). The table shows the bit numbering, the definition of the bit in each position (C for Check, D for Data, or P for Parity)

1	2	3	4	5	6	7	8	9	10	11	12	13
C1	C2	D1	C3	D2	D3	D4	C4	D5	D6	D7	D8	P
0	1	1	0	1	0	0	1	0	0	1	0	0

This data may be correct, may have a one-bit error, or may have a two-bit error. Which is it? If it's a 1-bit error, what is the correct data?

Recomputing the parity of the received data, I get a 1; hence, there is a one-bit error. Recomputing the check bits, I get

1	2	3	4	5	6	7	8	9	10	11	12	13
C1	C2	D1	C3	D2	D3	D4	C4	D5	D6	D7	D8	P
1	0	1	1	1	0	0	1	0	0	1	0	0

which differs in bits C1, C2 and C3. Therefore, there is a one-bit error in bit 7 (D4). Reversing that and recomputing, we get

1	2	3	4	5	6	7	8	9	10	11	12	13
C1	C2	D1	C3	D2	D3	D4	C4	D5	D6	D7	D8	P
0	1	1	0	1	0	1	1	0	0	1	0	0

<i>Points</i>	<i>Error</i>
-5	<i>Says parity correct, but one bit error</i>
-10	<i>No idea how syndrome calculated, got 1111</i>
-5	<i>No overall parity check</i>
-5	<i>Didn't correct data</i>
-2	<i>Said syndrome was 1, but corrected correct bit?</i>

A terminology note: the "syndrome" means the check bits that don't match the received data. A number of people used the word to mean the received check bits or the newly computed check bits... or a few people used it in ways I couldn't quite figure out what they meant! If you figured out it was bit 7 and fixed it, I didn't take points for the terminology error.