ECE M116C - CS M151B Computer Architecture Systems - UCLA

Quiz 3 (Upload it to Gradescope) - Open book and note

Fall 2022 Don't forget to sign in before you leave.

Notes:

- You are allowed to access any software, tools, online material, notes, and books during the quiz. But you are not allowed to text, chat, or post anything during the quiz.
- If you have any questions, the preferred method of communication is to DM me on Campuswire, but you can also raise your hand.
- You have 80 minutes to complete the quiz (until 1:30 PM).
- You can write your answers on a laptop, tablet, or paper. If your internet is currently not working, probably paper is a better option.
- Once you are finished, please upload your answers to Gradescope. You have an additional 10 minutes to upload your files. If you upload it after 1:30 PM, on Gradescope, it will show it as late, but you won't lose any points. Note that you won't be allowed to upload anything after 1:40 PM.
- If your internet doesn't work after a few tries, you can hand in your paper answers. If you wrote down your results in PDF, just take a screenshot and timestamp your file. We will let you upload it later.
- Good luck! :)

Q1 [40 points] **Memory Consistency:** *'Let's get confused!'*

Assume that we have a multi-core system with the following instruction sequences:

Initial Values: `Per Thread: x1=1, x2=0` `Global: flag = 0, Money = 0`

THREAD 1

```
x1 = Money;
acquire (lock);
Money = Money + 1;
release (lock);
```

THREAD 2

```
while(flag==0);
x1 = Money + x2;
```

THREAD 3

```
x2 = Money;
fence;
acquire (lock);
flag = 1;
Money = Money + 2;
release (lock);
```

a) If the memory model is Release Consistency, what are the possible values for `x1` in Thread 2? (Show the sequence by denoting numbers as T1.1, T1.2, etc. where Tx.y means the Yth instruction in Thread X).

(For each unique answer, show only one possible sequence that would create that value. We are only interested in possible values for x1 in Thread 2. You don't need to show the possible values for other values or other threads. And yes, you will get partial credit if you don't get them all!)

We are only interested in x1 of Thread 2 and what values it can end up with in different execution of the threads.

THREAD 1

```
T1.1 x1 = Money;
     acquire (lock);
T1.2 Money =Money + 1;
     release (lock);
```

THREAD 2

```
     while(flag==0);
T2.1 x1 = Money + x2;
```

THREAD 3

```
T3.1 x2 = Money;
     fence;
     acquire (lock);
T3.3 flag = 1;
T3.4 Money= Money + 2;
     release (lock);
```

So possible ways x1 can end up in different values are:

1. **T1.1, _T1lockAcquire_, T3.1, T1.2, _T1lockRelease_, _T3lockAcquire_, T3.3, T2.1…**
   x1 in Thread 2 = 1

2. **T1.1, _T1lockAcquire_, T3.1, T1.2, _T1lockRelease_, _T3lockAcquire_, T3.3, T3.4, T2.1…**
   x1 in Thread 2 = 3

3. **T1.1, T3.1, _T2lockAcquire_, T3.3, T2.1…**
   x1 in Thread 2 = 2

4. **T1.1, T3.1, _T2lockAcquire_, T3.3, T3.4, T2.1…**
   x1 in Thread 2 = 0

+8 pts for each x1 value, also, please note there can be more than one way to get to the four different x1 values. As long as they have gotten to one value they should get the full marks if flow of their trace makes sense. max score: 8+8+8+8 =32

-4 pts (negative) for each incorrect x1 values (i.e., if they found a value other than 0-3).

b) If we remove the locks (see below), add `barriers` and `fences` to guarantee `x1=3` in T2 at the end. (Same initial values and assumptions.)

THREAD 1

```
x1 = Money;
Money = Money + 1;
```

THREAD 2

```
while(flag==0);
x1 = Money + x2;
```

THREAD 3

```
x2 = Money;
fence;
flag = 1;
Money = Money + 2;
```

So from the answer of part a) we know we want the trace:
'T1.1, T3.1, T1.2, T3.3, T3.4, T2.1…' to occur.

THREAD 1

```
x1 = Money;
Money = Money + 1;
barrier1
```

THREAD 2

```
while(flag==0);
barrier2
x1 = Money + x2;
```

THREAD 3

```
x2 = Money;
barrier1
flag = 1;
Money = Money + 2;
barrier2
```

+4 pts per correct barrier. If they used fences, please manually check if the result is correct. Give 4 out 8 for attempting.

Q2. [20 points] **Cache Coherency:** *'Much less confusing, right'?*

Assume we have a MOESIF coherence protocol. Assume we have 4 cores and all cores are sharing the memory line `M[Z]`. Assume that the initial state for all of them is (I)nvalid.

Complete the following table (show the writebacks with *, and if one cache forwards a value to another cache show it by ^):

Access	P1	P2	P3	P4
R1				
R2				

W1				
R2				
W3				

Assuming MOESIF protocol:

Access	P1	P2	P3	P4
R1	E	I	I	I
R2	F^	S	I	I
W1	M	I	I	I
R2	O^	S	I	I
W3	I*	I	M	I

For each non-I state give them 2 pts (total of 7 non I). For each forwarding and/or writeback give 2 pts (total of three).

Q3 [30 points] **Virtual Address:** *'Let's NOT get physical!'*

Given a 32-bit operating system, 4 KB page size and 2 GB memory:

a) What is the max size of a VIPT 4-way set-associative cache with line size of 2 blocks?

For a no aliasing VIPT cache"
Ways in a set = 4.
Line size = 2 Blocks.
Block size = 4 Byte (as a word in a 32-bit system is 4 Bytes).
so Line size = 8 Bytes.
And given Page size = 4 kB = 2^12.

index width + block offset width <= page offset width

log2(sets) + log2(blocks) <= log2(page size)

sets = $\frac{cache\ size}{line\ size * associativity} = \frac{cache\ size}{8\ B * 4\ ways}$

log2($\frac{cache\ size}{8\ B * 4\ ways}$**) + log2(4B) <= log2(4kB)**

Cache size <= 32 kB.

10 pts for correct result.
Partial Credit: If figured out the Block size accurately: 2pts, correct equation 5 pts, correct calculation: 3 pts.

b) If we change the page size to 2 MB, fill the table below by putting increase (I), decrease (D), unchanged (U), or unknown (K) when comparing the new design with the old (i.e., part a). For each row, briefly explain why.

Metric	Change (I, D, U, K)	Why?
Storage overhead		
L1 VIPT Size		
Page Faults		
TLB miss rate		

Page size increased from 4 KB to 2 MB

Metric	Change (I, D, U, K)	Why?
Storage overhead	**D**	**Because of the increase in page size, we will have more page offset bits, and as result less PPNs and therefore fewer translations to save, reducing the page table size and therefore storage overhead**

		associated with storing said page table.
L1 VIPT Size	**I**	**Increase in page offset bits will result in the possibility of having more index+offset bits and therefore larger VIPT caches with no aliasing.**
Page Faults	**D**	**Larger pages means more addresses can be covered per page hence bigger coverage of the physical memory.**
TLB miss rate	**D / K**	**(We accept either).** **1- For some applications, larger pages means fewer entries in TLB hence higher chances of TLB hit. (decrease)** **2- For other group of applications, larger page size causes fragmentation and hence lower locality. This results in TLB pollution and higher miss rate.**

5 pts each. 2 points for attempt (i.e., if they provide an explanation that kinda make sense, but their answer is incorrect or if their answer is correct but their explanation doesnt make sense).

Q4 [10 points] **Short questions:** *'Size matters?!'*

a. Briefly describe how SMT and multi-core each improve performance?

SMT allows thread level parallelism for *independent* single-threaded applications. Multicore improves this further (independent application) as well as improving the performance of a multithreaded application.

5 pts total. 2 pts for mentioning thread level parallelism (TLP) in any of the two cases. 3 pts for mentioning independent vs. multithreaded for SMT vs. multicore.

b. Briefly describe the steps required for a CPU that is currently running program A, to read an input from keyboard during its execution?

The keyboard is an I/O device and works on the principles of Interrupts and Polling. First time you press a key it is usually an interrupt to the CPU. It receives the interrupt and halts the execution by changing the PC to a pre-specified part in the memory called Interrupt Service Routine (ISR). It then jumps to the relevant interrupt handler which reads the input from the I/O typically through memory mapped I/O or through dedicated registers. The values are forwarded back to the CPU and then the PC returns to original execution.

After the first interrupt is serviced the CPU activates the polling cycle for the keyboard input. Where it keeps checking the keyboard input channel (pre-specified memory address in case of memory mapped IO) in regular intervals while keeping on normal execution. If it ever sees a new input from the input channel it entertains that immediately.

5 pts. As long as they mention Interrupt and/or polling they should get the 2 pts. 2 pts for mentioning something about changing the PC. 1 pt for mentioning anything about memory-mapped I/O and/or dedicated registers.

Q5 [8 points] **Bonus:** *'Asking for a friend'*

a. [3 points] Is coffee or tea better? **wrong answers only!**

Shyandeep: At this point, I am a caffeine dependent organism, definitely need my cup(s) of coffee in the morning. I do love tea (especially chai) as well, but usually later in the day.

Everyone should get their 3 pts.

b. [5 points] REMINDER: Submit your course evaluation if you haven't done so (after the quiz)! 75% or more class participation gives you a 5% bonus for this quiz (or 8% if we get to 90%).

We had about 86% participation so give everyone 5 points here (i.e., total of 8 bonus points).