# CPSC 261 Sample Midterm 2
March 2016

Name: ____________________ Student ID: ____________________

Signature: ____________________

- You have 60 minutes to write the 5 questions on this examination.
  A total of 45 marks are available.
- **Justify all of your answers.**
- You are allowed to bring in one hand-written, double-sided 8.5 x 11in sheet of notes, and nothing else.
- Keep your answers short. If you run out of space for a question, you have written too much.
- The number in square brackets to the left of the question number indicates the number of marks allocated for that question. Use these to help you determine how much time you should spend on each question.
- Use the back of the pages for your rough work.
- **Good luck!**

| Question | Marks |
|---|---|
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| Total | |

UNIVERSITY REGULATIONS:

- Each candidate should be prepared to produce, upon request, his/her UBC card.
- No candidate shall be permitted to enter the examination room after the expiration of one half hour, or to leave during the first half hour of the examination.
- CAUTION: candidates guilty of any of the following, or similar, dishonest practices shall be immediately dismissed from the examination and shall be liable to disciplinary action.
  1. Having at the place of writing, or making use of, any books, papers or memoranda, electronic equipment, or other memory aid or communication devices, other than those authorised by the examiners.
  2. Speaking or communicating with other candidates.
  3. Purposely exposing written papers to the view of other candidates. The plea of accident or forgetfulness shall not be received.
- Candidates must not destroy or mutilate any examination material; must hand in all examination papers; and must not take any examination material from the examination room without permission of the invigilator.

[10] 1. Short answers

[3] (a) Multiplexing and aggregation are two techniques used for virtualization. How do they differ?

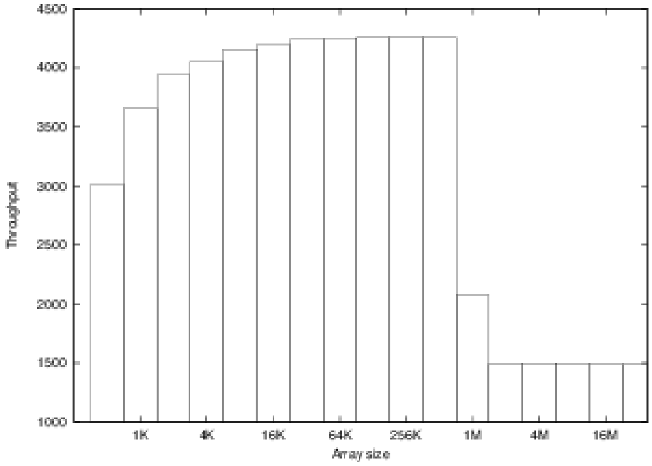
[2] (b) How can the parameters of a cache (block size, set size) be modified without making the cache bigger to take better advantage of temporal locality in a program? Explain your answer.

[2] (c) How can the parameters of a cache (block size, set size) be modified without making the cache bigger to take better advantage of spatial locality in a program? Explain your answer.

[3] (d) What is a deadlock? Explain one situation where it might occur.

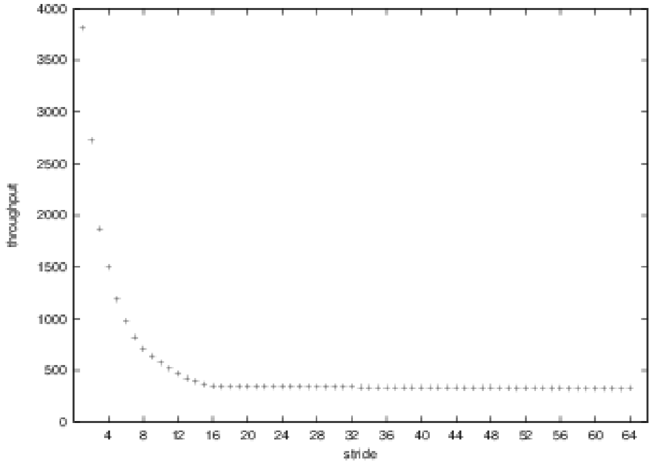
[6] 2. A memory mountain was obtained for an unspecified CPU. In this question, you will be asked to identify two features of the CPU, given a "slice" of that memory mountain. You **must** justify your answers.

[3] a. The following figure shows throughput as a function of the array size, for a fixed stride.



The largest size shown in the figure is larger than the size of the CPU's largest cache. What is the approximate size of that largest cache?

[3] b. The following figures shows throughput as a function of stride for an array that does **not** fit in the CPU's largest cache.



What is the approximate size of each cache line in bytes, knowing that `sizeof(int) == 4`?

[10] 3. In this problem, we will consider a cache that is 4-way set associative ($E = 4$), with 4-byte block size ($B = 4$) and 8 sets ($S = 8$). Suppose that the CPU accesses memory 1 byte at a time, uses 17-bit addresses ($m = 17$), and that the cache initially contains the following data (all values are in hexadecimal, B0 through B3 refer to the four bytes in each block):

| Set | Valid | Tag | B0 | B1 | B2 | B3 | Valid | Tag | B0 | B1 | B2 | B3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 5B3 | CC | 28 | 41 | 3D | 1 | 6FA | 45 | CD | 64 | 6D |
| | 0 | CFC | 9E | 38 | 19 | C9 | 1 | A45 | E2 | BB | 2D | 9F |
| 1 | 1 | 126 | AB | D9 | 85 | CE | 1 | CC2 | 3D | 84 | F1 | 75 |
| | 1 | 431 | 7F | D5 | A1 | C1 | 0 | 1DA | 26 | 5F | 9B | 5B |
| 2 | 1 | 769 | B6 | 41 | 96 | 67 | 0 | 59C | 50 | 39 | C8 | 48 |
| | 0 | 908 | 70 | 3E | 0A | A4 | 1 | FA9 | 25 | FC | 06 | 34 |
| 3 | 1 | 7A4 | F5 | 20 | 7B | B7 | 0 | 602 | CD | C3 | C6 | EF |
| | 1 | 99C | 01 | C2 | DE | 8C | 1 | 4F0 | 39 | 6F | 16 | 6D |
| 4 | 0 | B29 | 79 | BE | 58 | 3D | 1 | A4E | 65 | 58 | 3F | 0E |
| | 1 | 94F | 6A | 87 | 68 | 09 | 1 | FD8 | D8 | D8 | 9C | F9 |
| 5 | 1 | 409 | E3 | 49 | 28 | 1A | 0 | 806 | 54 | 13 | 8A | 9E |
| | 1 | AD9 | 94 | F0 | 82 | EF | 1 | 650 | 75 | CA | 28 | E3 |
| 6 | 0 | F1C | 1A | 71 | 40 | CD | 0 | 22D | EA | 3F | 85 | 18 |
| | 1 | 506 | A7 | 4C | 88 | C1 | 1 | 4BE | 9A | 17 | D7 | 58 |
| 7 | 1 | 32D | F5 | 11 | 9A | 26 | 1 | C3B | 3D | F7 | 20 | 9E |
| | 1 | 84A | 58 | 84 | EB | 46 | 1 | 4E4 | 2E | 38 | 80 | 33 |

[2] (a) The following diagram shows the format of an address (one bit per box). Write in each box which field the bit belongs to: CO (the cache block offset), CI (the cache set index) and CT (the cache tag).

| 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | |

[8] (b) The following table contains, for four **read** operations:

i. The binary representation of the address using 17 bits.
ii. Which set will be searched to find the data.
iii. The tag that will be compared against the cache lines' tags.
iv. Whether it's a cache hit or a cache miss.
v. If it's a cache hit, what value was read from the cache.

Fill in all entries in this table (except for the value read if the access for that row is a cache miss). A binary to hexadecimal conversion table is included on the last page of this test.

| Address (binary) | Set | Tag | Hit or Miss? | Value read |
|---|---|---|---|---|
| 0 1100 0000 0100 1110 | | | | |
| 0 1000 0110 0010 0111 | | | | |
| 0 1010 1101 1101 1011 | | | | |
| | 4 | C4B | Miss | |

[13] 4. Threads Synchronization

[2] a. Under which conditions does a *race condition* occur? Be as precise as you can.

[2] b. In class, we discussed three correct implementations of our bounded buffer example: one that uses locks only, one that uses locks and condition variables, and one that uses locks and semaphores. What advantage do the second and the third implementations have over the first one?

[3] c. Consider the first implementation of our bounded buffer example:

```
while (buf->in - buf->out == N) {
    pthread_mutex_unlock(&lock);
    usleep(random() % LITTLESTALL);
    pthread_mutex_lock(&lock);
}
```

Why is the lock release before the call to `usleep` and reacquired afterwards?

[3] d. Consider now the second implementation of our bounded buffer example (this is taken from our `send` function):

```
while (buf->in - buf->out == N) {
    pthread_cond_wait(&forspace, &lock);
}
```

The call to `pthread_cond_wait` will terminate (and return to the `send` function) under two situations. Which ones?

[3] e. Why did we not write

```
if (buf->in - buf->out == N) {
    pthread_cond_wait(&forspace, &lock);
}
```

instead?

[6] 5. A CPU has a 2-way set associative ($E = 2$) cache, with 16-byte block size ($B = 16$), 8 sets ($S = 8$), and a least recently used replacement policy. Assume that `sizeof(int)` is 4 and that we have the following C declaration:

```
int a[4][32];
```

What will be the approximate miss rate for each of the following loops? Justify your answers!

[3] a.

```
for (i = 0; i < 32; i++)
    for (j = 0; j < 4; j++)
        sum += a[j][i];
```

[3] b.

```
for (j = 0; j < 4; j++)
    for (i = 0; i < 32; i++)
        sum += a[j][i];
```

| Hex | Binary | Hex | Binary |
|---|---|---|---|
| 0 | 0000 | 8 | 1000 |
| 1 | 0001 | 9 | 1001 |
| 2 | 0010 | A | 1010 |
| 3 | 0011 | B | 1011 |
| 4 | 0100 | C | 1100 |
| 5 | 0101 | D | 1101 |
| 6 | 0110 | E | 1110 |
| 7 | 0111 | F | 1111 |