15-418/618 Exam 1

Jiaqiang Ruan

TOTAL POINTS

46.25 / 54

QUESTION 1

Multiple Choice 11 pts

1.1 1A 1 / 1

✓ + **1 pts** Correct

+ **0 pts** Incorrect

1.2 1B 1 / 1

✓ + **1 pts** Correct

+ **0 pts** Incorrect

1.3 1C 1 / 1

✓ + **1 pts** Correct

+ **0 pts** Incorrect

1.4 1D 1 / 1

✓ + **1 pts** Correct

+ **0 pts** Incorrect

1.5 1E 0 / 1

+ **1 pts** Correct

✓ + **0 pts** Incorrect

1.6 1F 1 / 1

✓ + **1 pts** Correct

+ **0 pts** Incorrect

1.7 1G 1 / 1

✓ + **1 pts** Correct

+ **0.75 pts** 3/4 correct (should be a, b, c)

+ **0.5 pts** 2/4 correct (should be a, b, c)

+ **0.25 pts** 1/4 correct (should be a, b, c)

+ **0 pts** 0/4 correct (should be a, b, c)

1.8 1H 1 / 1

✓ + **1 pts** Correct

+ **0.75 pts** 3/4 correct (should be a, b)

+ **0.5 pts** 2/4 correct (should be a, b)

+ **0.25 pts** 1/4 correct (should be a, b)

+ **0 pts** Incorrect (should be a, b)

1.9 1I 1 / 1

✓ + **1 pts** Correct

+ **0.75 pts** 3/4 correct (should be a, c, d)

+ **0.5 pts** 2/4 correct (should be a, c, d)

+ **0.25 pts** 1/4 correct (should be a, c, d)

+ **0 pts** Incorrect

1.10 1J 0.25 / 1

+ **1 pts** Correct

+ **0.75 pts** 3/4 correct (should be a, c)

+ **0.5 pts** 2/4 correct (should be a, c)

✓ + **0.25 pts** **1/4 correct (should be a, c)**

+ **0 pts** Incorrect

1.11 1K 1 / 1

✓ + **1 pts** Correct

+ **0.75 pts** 3/4 correct (should be a, b)

+ **0.5 pts** 2/4 correct (should be a, b)

+ **0.25 pts** 1/4 correct (should be a, b)

+ **0 pts** Incorrect

QUESTION 2

ILP/SIMD 9 pts

2.1 2A DF Diagam 3 / 3

✓ + **2 pts** **Correct data flow diagram**

+ **1 pts** Partial data flow diagram

✓ + **1 pts** **Mention x is the only loop-carried dependency**

+ **0.5 pts** Partial loop-carried dependency

2.2 2B OOO 1 / 2

✓ + 1 pts **Mention the computation of a and b could process in parallel**

+ 1 pts Mention this would require either two multipliers or a pipelined multiplier

+ 0 pts Incorrect

2.3 2C.1 WC Speedup 2 / 2

✓ + 2 pts **Correct**

+ 1 pts Partial

+ 0 pts Incorrect

2.4 2C.2 BC Speedup 2 / 2

✓ + 2 pts **Correct**

+ 1 pts Partial

+ 0 pts Incorrect

QUESTION 3

GPU/CUDA 12 pts

3.1 3A Ordering 2 / 2

✓ + 2 pts **Correct**

+ 0 pts Incorrect

blocks > warps > threads

3.2 3B Local synchthreads 2 / 2

✓ + 2 pts **Correct. Something along the lines of "CUDA blocks do not get swapped in and out of execution like CPU threads but get run to completion. Blocks that are not yet scheduled would never reach __syncthreads() as the scheduled blocks would wait at the barrier and never terminate." Also accepted similar answers to, "threads/warps within a block run concurrently, blocks themselves do not"**

+ 1 pts Mentions some way blocks are run independently from each other, or mentions some kind of difficulty communicating between blocks.

+ 0 pts Incorrect

3.3 3C Global synch 0 / 2

+ 2 pts Correct. Interleaving of block execution, similar to how CPU threads run concurrently.

+ 1 pts Limiting number of blocks launched to fit on GPU SMs. While totally limiting the capability of the GPU, a global barrier would be possible as all blocks would be running simultaneously.

✓ + 0 pts **Incorrect.**

3.4 3D(a) synchthreads #1 1 / 1

✓ + 1 pts **Correct**

+ 0 pts Incorrect

3.5 3D(b) synchthreads #2 1 / 1

✓ + 1 pts **Correct**

+ 0 pts Correct

3.6 3D(c) synchthreads #3 1 / 1

✓ + 1 pts **Correct**

+ 0 pts Incorrect

3.7 3D(d) reduce memory accesses 3 / 3

✓ + 3 pts **Copies to local/shared memory**

+ 0 pts Incorrect

QUESTION 4

Cache Design & Operation 10 pts

4.1 4A.1 Full bit vector 2 / 2

✓ + 2 pts **(M/64 * P) / 8 bytes**

+ 0 pts Incorrect

4.2 4A.2 Limited pointer scheme 2 / 2

✓ + 2 pts **(3 log(P) * M/64) / 8 bytes, (log base 2, not required to specify)**

+ 0 pts Incorrect

4.3 4B.1 Producer Phase 3 / 3

✓ + 3 pts **Correct**

+ 2 pts Within small constant factor of correct answer

+ 2 pts Should only show for single execution of phase

+ **0 pts** Not within constant factor of correct answer
+ **1 pts** Within constant factor of correct answer
+ **1 pts** Miss fact that there are 16 X 500 writes
+ **1 pts** Only one bus transaction per write
+ **1 pts** Should show for execution of single phase and one transaction/write

4.4 4B.2 Consumer Phase 3 / 3

✓ + **3 pts** **Correct**
+ **0 pts** Not within constant factor of correct result
+ **1 pts** Only single cache line
+ **2 pts** Should only consider access to array data
+ **2 pts** One transaction per thread
+ **1 pts** Right answer, but poor explanation
+ **0 pts** Not sure what is meant by "N"

QUESTION 5

12 pts

5.1 5A N = 64 and P = 64 3 / 3

✓ + **1 pts** **(a) Correct**
✓ + **1 pts** **(b) Correct**
✓ + **1 pts** **(c) Correct**
+ **0 pts** Incorrect

5.2 5B Arbitrary N and P 5 / 5

✓ + **2 pts** **N^3/P or (N/q)^3**
✓ + **2 pts** **6(N^2/P^{2/3}) or 6(N/q)^2**
✓ + **1 pts** **N/(6P^{1/3}) or N/(6q)**
- **1 pts** Dropped constant
+ **0 pts** Incorrect

5.3 5C P' = 512 and N' = N 1 / 4

✓ + **1 pts** **(a) Correct**
+ **1.5 pts** (b) Justify 8x
+ **1.5 pts** (b) Justify 4x
+ **0 pts** Incorrect

gradescope

Full Name: Jiaqiang Ruan

Andrew Id: ~~Ja~~ Jruan

15-418/618, Spring 2020

Exam 1

February 26, 2020

Instructions:

- Write your answers in the space provided for the problem. If your work gets messy, please clearly indicate your final answer.
- The exam has a maximum score of **54** points.
- The problems are of varying difficulty. The point value of each problem is indicated. Pile up the easy points quickly and then come back to the harder problems.
- This exam is CLOSED BOOK, CLOSED NOTES (with the exception of your one sheet of notes.)

Problem	Your Score	Possible Points
1		11
2		9
3		12
4		10
5		12
Total		54

Multiple Choice Answer

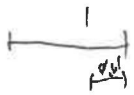
Problem 1. (11 points):

Problems that state "circle all that apply" may have any number of correct answers (including zero). The other problems have unique answers.

A. Super-linear scaling is often the result of

(a) Lower arithmetic intensity.
(b) False sharing.
(c) Instruction-level parallelism.
(d) Data fitting in the cache.

B. Suppose there is a program that is 90% parallelizable. What is the maximum possible speedup from parallelism given infinite processors?



(a) 100×
(b) 10×
(c) 50×
(d) 8×

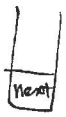
C. CUDA kernels run in what fashion?

(a) SIMD
(b) SPMD
(c) MISD
(d) SISD

D. In comparison to a CPU, a GPU spends most of its chip area on:

(a) Control
(b) Memory
(c) ALUs
(d) I/O

E. Which one of the following statements is incorrect?




```
for (int i=0; i<N; i++) {
    cilk_spawn foo(i);
}
cilk_sync;
```

(a) A "child stealing" policy could quickly fill the task queues at the beginning.
(b) There might be a thread that does not execute any instance of function `foo`.
(c) In the greedy stealing policy, all threads always attempt to steal if they are idle.
(d) If there is not enough parallelism, the programmer could rewrite the program to increase the granularity.

F. Which one of the following statements is incorrect?

(a) With shared-memory parallelism, processors communicate implicitly via load/store.

(b) With message-passing parallelism, sending and receiving ghost cells might lead to deadlock.

(c) In the Ocean Simulation Problem, the Blocked assignment requires more data to be communicated between processors than the Interleaved assignment.

(d) With message-passing parallelism, it is better to send an entire row than to send individual elements one at a time.

G. Which of the following states in a MESI cache coherence protocol allow a processor to *read* data without additional communciation to other cache controllers? Circle all that apply.

(a) M
(b) E
(c) S
(d) I

H. Which of the following states in a MESI cache coherence protocol allow a processor to *write* data without additional communication to other cache controllers? Circle all that apply.

(a) M
(b) E
(c) S
(d) I

I. Which of the following could a programmer realistically launch a couple thousand of on a Gates cluster machine? Circle all that apply.

(a) CUDA threads
(b) pthreads
(c) ISPC tasks
(d) cilk threads

J. Suppose our application involves testing N points for membership in the Mandelbrot set and recording the number of iterations tested in an array. Which of the following scaling conventions decrease the amount of work per processing element? Circle all that apply.

explain = choose d.

(a) Strong
(b) Weak
(c) Problem-constrained
(d) Time-constrained

K. Which of the following forms of CPU parallelism can improve the performance of single-threaded programs? Circle all that apply.

(a) SIMD
(b) Instruction-level
(c) Multicore
(d) Simultaneous multithreading

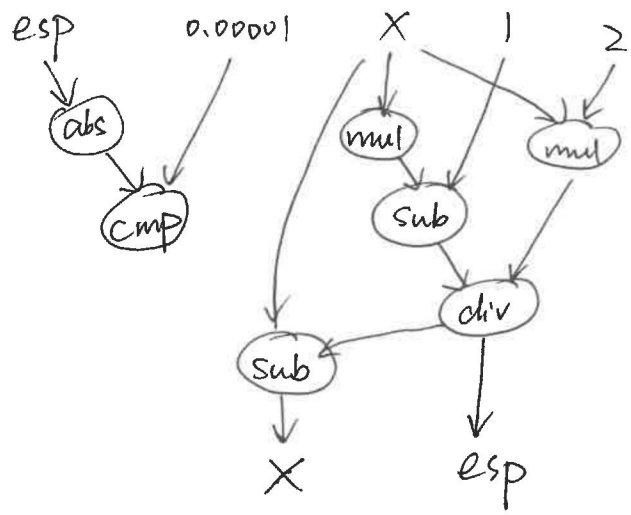
ILP/SIMD

Problem 2. (9 points):

We want to approximate root values of a function $f(x)$. A good way to do so is by using Newton's method, shown below with $f(x) = x^2 - 1$, beginning with an initial value of x_0 and "converged" when $\epsilon < 0.0001$.

```
double newton(double x0) {
    double x = x0;
    double eps = 1;
    while (abs(eps) >= 0.00001) {
        double a = x * x - 1;
        double b = 2 * x;
        eps = a / b;
        x = x - eps;
    }
    return x;
}
```

A. (3 Points) Draw the data flow diagram for the instructions in the while loop. Annotate the loop-carried dependencies.



The loop dependencies are:

X → mul → sub → div → sub → X.

X → mul → dive → sub → X

X → sub → X

B. (2 points) You have an old processor that uses pipelining, but it executes instructions in program order. You see a deal on a new processor that can take advantage of greater instruction-level parallelism (ILP) via out-of-order execution. (Neither does any speculative execution.) What part of this function could make use of this added form of ILP, and what sort of execution unit(s) would be required to do so?

The calculation of a and b. can be run in out-of-order manner.

The float-multiplication unit would be require to do so.

C. (4 Points) Now we would like to run Newton's method on an array of $N = 16$ powers of 16 using the following ISPC loop on a machine with a 4-wide SIMD vector unit.

```
foreach (i = 0 ... N) {
    output[i] = newton(pow(16.0, input[i]);
}
```

$0 \Rightarrow x = 1 \Rightarrow iter = 1$
$1 \Rightarrow x = 16 \Rightarrow$

$2^0, 2^4, 2^8, \ldots$

We are interested in running Newton's method on 16^i for i ranging from 0 to 15. *Note* that Newton's method will converge at different rates for different inputs. The graph below shows the required iterations for solving Newton's method on input 2^i for different values of i. Note also that $16^i = 2^{4 \cdot i}$.

16^i

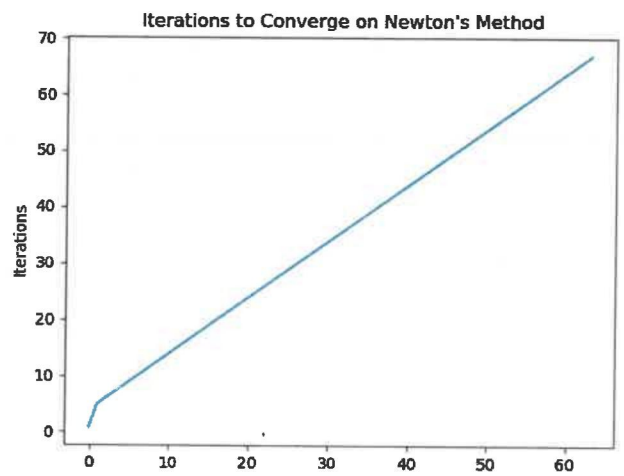


Figure 1: Iterations to Converge on Newton's Method for input 2^i, as a function of i

How could we order the order the data in the array `input` to get the worst relative speedup with SIMD?

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
input[i]	15	11	7	3	14	10	6	2	13	9	5	1	12	8	4	0

How could we order the order the data in the tabular `input` to get the best relative speedup with SIMD?

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
input[i]	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

GPU and CUDA

Problem 3. (12 points):

A. (2 Points) Order threads, warps, and blocks with largest granularity first and smallest granularity last.

blocks, threads, warps

B. (2 points) Explain why CUDA's `__syncthreads()` only acts as a barrier for threads within a block and cannot be used as a global barrier for all threads across all blocks.

Because threads are ~~grouped~~ launched within block and grouped into warps to run simultaneously. A block has its own ~~private~~ local memery and __syncthreads() is used to barrier threads in the same block. Threads in different block have ~~di~~ seperated control flow.

C. (2 points) What fundamental capability would be required to implement a global barrier synchronization in CUDA that is lacking in current NVIDIA GPUs?

The message passing ~~ex~~ mechanism between different blocks ~~are~~ is needed.

(Or a global ~~ex~~ lock mechanism)

D. (6 Points) Below is some code for a CUDA kernel of a peculiar function. The code first shuffles the elements, then accumulates a local sum in every 4th element.

```
__global__ void peculiar(int N, float* in_out) {
    // Assume blockDim.x is multiple of 4
    int index = blockDim.x * blockIdx.x + threadIdx.x;

    float temp_priv = in_out[index];

    __syncthreads(); // #1

    if (index < N) {
      /* Permute odd and even elements */
      if (index % 2 == 0)
          in_out[index+1] = temp_priv;
      else
          in_out[index-1] = temp_priv;
    }

    __syncthreads(); // #2

    if (index < N && index % 4 == 0) {
      in_out[index] +=
            (in_out[index+1] + in_out[index+2] + in_out[index+3]);
    }

    __syncthreads(); // #3
}
```

in_out[0] → [1]

in_out[1] → [0].

(a) In one sentence, explain what would happen if we removed "__syncthreads()" #1?

temp_priv gets the flase value after permuting, instead of the original one.

(b) In one sentence, explain what would happen if we removed "__syncthreads()" #2?

Values that haven't been permutated will be add to "in_out[index]"

(c) In one sentence, explain what would happen if we removed "__syncthreads()" #3?

No difference.

(d) Explain briefly how we could reduce the amount of access to global memory by this kernel.

use threads to copy the in_out array into shared memeory first, and then update it back. Use _syncthreads() to make sure the "copy" and "update" finish or start before other calculation.

Coherent Cache Design and Operation

Problem 4. (10 points):

Part A

$\Rightarrow$ p bits.

Consider a multicore machine with P cores and M bytes of physical memory. It uses a directory-based cache coherence protocol, where cache blocks are 64 bytes. For the following two schemes, give the total number of bytes required to store the "presence state." (You need not consider any dirty or overflow bits.) Express your answers as formulas in terms of P and M.

1. (2 points) Full bit vector:

$$\frac{M}{64} \times P \text{ bits} = \frac{MP}{512} \text{ bytes}$$

2. (2 points) Limited pointer scheme with 3 pointers:

$$3 \times \frac{M}{64} \times \log_2 P \text{ bits} = 3 \times \frac{M \log_2 P}{512} \text{ bytes}$$

Part B

The following problem concerns a multi-threaded program running on a 16-core processor. Assume the following:

- Each core has a single-level cache with a block size of 64 bytes.
- The processor uses a snoopy bus to maintain cache coherence.
- The caches are very large, and so conflict misses can be ignored.

The program has the following global declarations and variables

```
#define TCOUNT 16
#define N 64
int rounds = 2000;
int tries = 500;
unsigned char data[N];   // Assume address is multiple of 64
int tsum[TCOUNT];
```

It makes use of the following thread routine:

```
void *threadP(void *args) {
    size_t id = (size_t) args;
    for (int r = 0; r < rounds; r++) {

        /* Producer Phase */
        for (int try = 0; try < tries; try++) {
            int idx = random() % N;
            data[idx] = id;
        }
        pthread_barrier_wait(&barrier);

        /* Consumer Phase */
        int sum = 0;
        for (int try = 0; try < tries; try++) {
            int idx = random() % N;
            if (data[idx] == id)
                sum++;
        }
        tsum[id] += sum;
        pthread_barrier_wait(&barrier);

    }
    return NULL;
}
```

The threads are spawned using the following code

```
    for (size_t t = 0; t < TCOUNT; t++) {
        pthread_create(&tid[t], NULL, threadP, (void *) t);
    }
```

1. (3 points) Consider the Producer Phase. What is the worst case total number of bus transactions that could occur during one execution of the phase by all threads due to accesses to array `data`? Justify your answer.

$16 \times 500 = 800$

data is an array with 64 unsigned char and fits into one single cache line exactly. So 16 processors will acquire write on this cache line. In worst case, each processor requires tires time if the cache line is not in its priva cache.

2. (3 points) Consider the Consumer Phase. What is the worst case total number of bus transactions that could occur during one execution of the phase by all threads due to accesses to array `data`? Justify your answer.

16

Consumer phase only need the permission for read. So the wrost case is that all processors need one valid copy of the cach line. So it needs 16 ~~trans~~ transaction.

Scaling

Problem 5. (12 points):

You have just joined the staff of the UN Intergovernmental Panel on Climate Change (IPCC) to help them implement their global climate model. This model represents the atmosphere using a grid containing $N \times N \times N$ grid locations.

For all of the following questions, consider what a *single* processor would do. If some processors require more communication than others, give the value for the ones with the maximum requirement.

A. (3 points) Their current simulation has $N = 64$ (i.e., the grid is $64 \times 64 \times 64$) and its implementation runs on $P = 64$ processors. Each processor handles a block of $16 \times 16 \times 16$ grid points.

(a) How many grid points does a processor compute for *one* iteration?

$$\frac{64 \times 64 \times 64}{64} = 4096$$

(b) How many data values does a processor need to retrieve from other processors for *one* iteration?

$$16 \times 16 \times 6 = 1536$$

(c) What is the arithmetic intensity?

$$\frac{4096}{1536} = \cancel{2.667}\ \frac{8}{3} \approx 2.67$$

B. (6 points) Answer the following as a function of (linear dimension) grid size N and number of processors P. (Hint: You may find it helpful to use $q = P^{1/3}$.)

(a) How many grid points does a processor compute for *one* iteration?

$$\frac{N^3}{P}$$

(b) How many data values does a processor need to retrieve from other processors for *one* iteration?

$$6 \times \left(\frac{N}{q}\right)^2 = \frac{6N^2}{P^{\frac{2}{3}}} \quad \text{or} \quad \frac{N^2}{P^{\frac{2}{3}}}$$

(c) What is the arithmetic intensity?

$$\frac{N^3}{P} \Big/ \frac{6N^2}{P^{\frac{2}{3}}} = \frac{N}{6P^{\frac{1}{3}}} \quad \text{or} \quad \frac{N}{P^{\frac{1}{3}}}$$

C. (4 pts) The rapid response team wants to make the model run faster by using $P' = 512$ processors, keeping the problem size the same $N' = N$.

(a) Of the three types of scaling described in class, what type of scaling is this?

problem-constrained scaling

(b) Your colleague expects to get somewhere between $4\times$ and $8\times$ speedup on the new machine. Justify their reasoning *using the equations you derived in part B.*

$N' = N$

$P' = 8P$

~~Bec~~ ~~I = A~~.

Therefore

$$A' = \frac{N'}{P'^{\frac{1}{3}}} = \frac{N}{2P} = \frac{1}{2}A$$

The speedup is only 2x