University of Pennsylvania
Department of Electrical and System Engineering
System-on-a-Chip Architecture

ESE532, Fall 2021 Midterm Solutions Wednesday, October 6

- Exam ends at 11:45AM; begin as instructed (target 10:15AM)
  Do not open exam until instructed.
- Problems weighted as shown.
- Calculators allowed.
- Closed book = No text or notes allowed.
- Show work for partial credit consideration. All answers here.
- Unless otherwise noted, answers to two significant figures are sufficient.
- Sign Code of Academic Integrity statement (see last page for code).

____

I certify that I have complied with the University of Pennsylvania's Code of Academic Integrity in completing this exam.

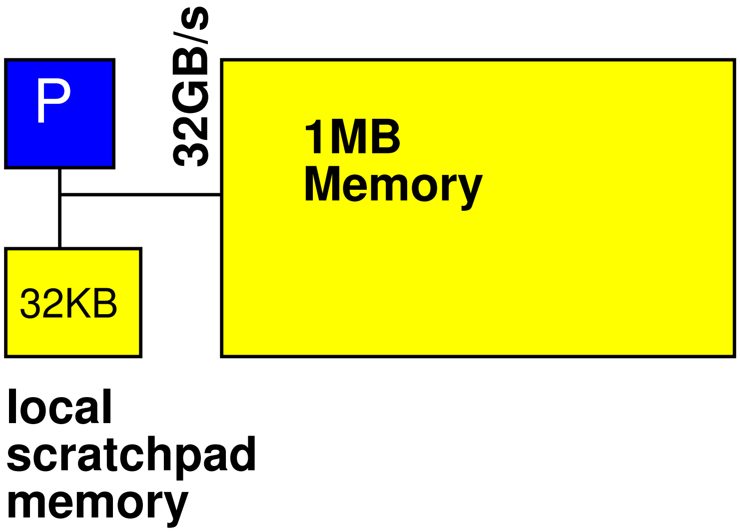
Name: Solutions

1	2a	2b	3	4	5	6	7	8	Total
10	5	5	10	10	10	20	10	20	100

Consider the following (very simplified) code to localize a point based on a vector of readings.

```
#define REF_SOURCES 100
#define NUM_KNOWN_POINTS 200
#define NUM_NEIGHBORS 5
#include <stdlib.h>

int known_points[NUM_KNOWN_POINTS][REF_SOURCES];
int neighbor_db[NUM_KNOWN_POINTS][NUM_NEIGHBORS];
int known_points_x[NUM_KNOWN_POINTS];
int known_points_y[NUM_KNOWN_POINTS];

int distance(int *v1,int *v2) {
  int res=0;
  for (int i=0;i<REF_SOURCES;i++) // loop H
    res+=abs(v1[i]-v2[i]);
  return(res);
}
int main() {
  int source_vector[REF_SOURCES];
  int pdist[NUM_KNOWN_POINTS];
  int ndist[NUM_NEIGHBORS];
  int neighbor[NUM_NEIGHBORS];
  int mindist, minref;

  int x=0;
  int y=0;
  int old_x=0;
  int old_y=0;
  read_known_points(known_points,known_points_x,known_points_y);

  while (1) { // loop A
      read_sources(source_vector); // for simplicity assume 0
               // maybe loaded into main memory by a separate processor
      for (int i=0;i<NUM_KNOWN_POINTS;i++) // loop B
          pdist[i]=distance(known_points[i],source_vector);
      for (int i=0;i<NUM_KNOWN_POINTS;i++) { // loop C
          if (mindist>pdist[i]) {
              mindist=pdist[i];
              minref=i;
            }
        }
      for(int j=0;j<NUM_NEIGHBORS;j++) // loop D
        neighbor[j]=neighbor_db[minref][j];
      for(int j=0;j<NUM_NEIGHBORS;j++) // loop E
        ndist[j]=distance(known_points[neighbor[j]],source_vector);
      int totdist=0;
      for(int j=0;j<NUM_NEIGHBORS;j++) // loop F
        totdist=ndist[j]+totdist;
      old_x=x;
      old_y=y;
      x=0;
      y=0;;
      for(int j=0;j<NUM_NEIGHBORS;j++) { // loop G
          x+=known_points_x[neighbor[j]]*((totdist-ndist[j])/((NUM_NEIGHBORS-1)*totdi
;
          y+=known_points_y[neighbor[j]]*((totdist-ndist[j])/((NUM_NEIGHBORS-1)*totdi
;
        }
      int dx=x-old_x;
      int dy=y-old_y;
      write_output(x,y,dx,dy); // assume 4 writes to main memory
    }
  return(0); // won't reach here
}
```

We start with a baseline, single processor system as shown.



- For simplicity throughout, we will treat non-memory indexing adds (subtracts count as adds), compares, min, max, abs, divides, and multplies as the only compute operations. We'll assume the other operations take negligible time or can be run in parallel (ILP) with the adds, multiplies, and memory operations. (Some consequences: You may ignore loop and conditional overheads in processor runtime estimates; you may ignore computations in array indecies.)
- Baseline processor can execute one multiply, divide, compare, min, max, abs, or add per cycle and runs at 1 GHz.
- Data can be transfered from the 1MB main memory at 32 GB/s when streamed in chunks of at least 256B. Assume `for` loops that only copy data can be auto converted into streaming operations.
- Non-streamed access to the main memory takes 10 cycles.
- Baseline processor has a local scratchpad memory that holds 32KB of data. Data can be streamed into the local scratchpad memory at 32 GB/s. Non-streamed accesses to the local scratchpad memory take 1 cycle.
- By default, all arrays live in the main memory.
- Arrays `ndist` and `neighbor` live in local scratchpad memory.
- Assume scalar (non-array) variables can live in registers.
- Assume all additions are associative.
- Assume comparisons, adds, min, max, divide and multiplies take 1 ns when implemented in hardware accelerator, so fully pipelined accelerators also run at 1 GHz. A compare-mux operation can also be implemented in 1 ns.
- Data can be transfered to accelerator local memory at the same 32 GB/s when streamed in chunks of at least 256B.

1. Simple, Single Processor Resource Bounds

   Give the single processor resource bound time for compute operations and memory access for each loop inside loop A (and non-loop code at end) and the total bound for loop A.

loop	Compute	Memory
B	H: 100 x (subtract, add, abs = 3)= 300 200xH: 60,000	H: 100 x (v1[] (known_points[i]), v2[] (source_vector) = 20) 2000 200x(H+10)(pdist[i] = 10)=402,000
C	200 x (> = 1) = 200	200x(pdist[i] = 10)=2000
D	0	5 x (neighbor[j] = 1, neighbor_db[minref][j] = 10) = 55
E	5xH = 1500	5x(H+ (neighbor[j], ndist[j] = 2) = 10,010
F	5 (add = 1) = 5	5 (ndist[i] = 1) = 5
G	5x2x(+,*,-,-,*,/)=60	5x2x(kown_points_{x,y}[j], neighbor[j],ndist[j] = 12) = 120
non-loop code after G	2 (-)	40
A	61,767	414,230

2. Based on the simple, single processor mapping from Problem **??**:

   (a) What loop is the bottleneck? (circle one)

   B

   C

   D

   E

   F

   G

   (b) What is the Amdahl's Law speedup if you only accelerate the identified function?

   $\frac{474192}{13992} = 33.8 \approx 34$

3. Parallelism in Loops

   (a) Classify the following loops as data parallel or not? (loop bodies could be executed concurrently)

   (b) Explain why or why not?

Loop	Data Parallel?	Why or why not?
B	Y	all pdist[i] calculations independent
C	N	sequential max across elements, keeping first index with max value (could also be reduce with cleverness)
D	Y	each copy independent
E	Y	all ndist[i] calculations independent
F	N (Y)	Add accumulating loop iterations makes this a reduce (we will classify separartely later)
G	N (Y)	Add accumulating loop iterations makes this a reduce (we will classify separartely later)
H	N (Y)	Add accumulating loop iterations makes this a reduce (we will classify separartely later)

4. What is the critical path for the body of loop A?

B	all distance (H) in parallel read v1,v2 in parallel 10 sub, abs +2 associative reduce add $\log_2(100) = 7$ pdist write 10	29
C	read all pdist[i] in parallel – 10 serial compare – 200	210
D	all read/write parallel – 11	11
E	all distance (H) in parallel read v1, v2, sub, abs – 12 associative reduce add $\log_2(100) = 7$ ndist write parallel – 1	20
F	read ndist parallel - 1 associative reduce add $\log_2(5) = 3$	4
G	read neighbor[j], ndist[j], subtracts - 1 read known_points_x,y (also 1st multiply, divide) - 10 final multiply - 1 associative reduce adds $\log_2(5) = 3$	15
after	subtracts in parallel - 1 writes in parallel - 10	11
Total		300

5. Rewrite the body of `loop A` to minimize the memory resource bound by exploiting the scratchpad memory and streaming memory operations.
   - Annotate what arrays live in the local scratchpad
   - Account for total memory usage in the local scratchpad (use provided table)
   - Provide your modifications to the code.
     - Use **for** loops that only copy data to denote the streaming operations
   - Estimate the new memory resource bound for your optimized `loop A`.

Variable	Size (Bytes)
neighbor	20
ndist	20
pdist	800
source_vector	400
known_points_tmp	400

Put pdist and source_vector in small memory; stream each known_points array into a temporary (known_points_tmp) in small memory before call distance.

```
for (int i=0;i<NUM_KNOWN_POINTS;i++) // loop B
    for (int j=0;j<REF_SOURCES;j++) // streaming operation
        known_points_tmp[j]=known_points[i][j];
    pdist[i]=distance(known_points_tmp,source_vector);
```

Can also stream for ndist, but big benefit is pdist, above.

```
for (int i=0;i<NUM_KNOWN_POINTS;i++) // loop E
    for (int j=0;j<REF_SOURCES;j++) // streaming operation
        known_points_tmp[neighbor[i]]=known_points[i][j];
    ndist[i]=distance(known_points_tmp,source_vector);
```

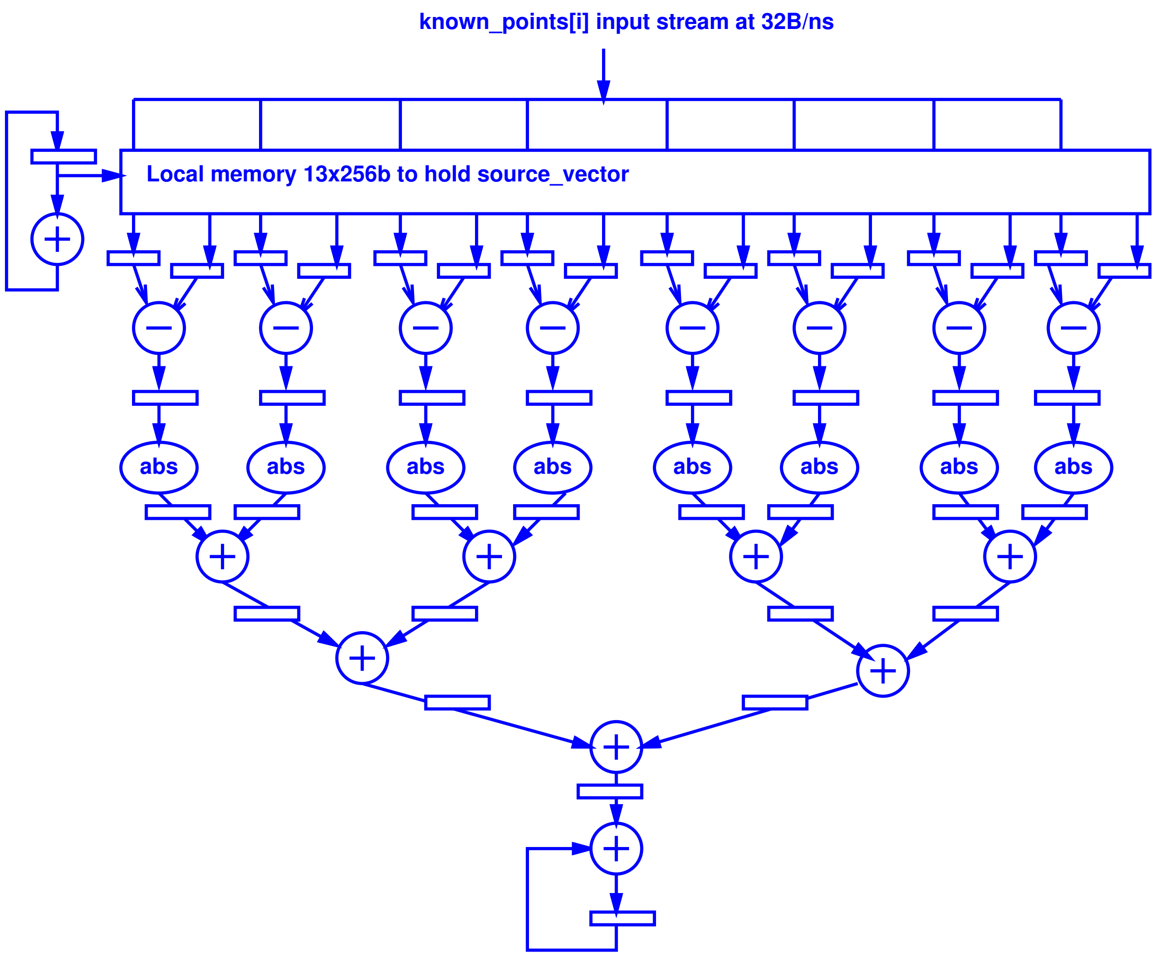
(This page intentionally left mostly blank for answers.)

loop	Compute	Memory
B	H: 100 x (subtract, add, abs = 3)= 300 200xH: 60,000	H: 100 x (v1[], v2[] = 2) 200 200x(H+13+1)(stream 400/32=13, pdist[i] = 1)=42,800
C	200 x (> = 1) = 200	200x(pdist[i] = 1)=200
D	0	5 x (neighbor[j] = 1, neighbor_db[minref][j] = 10) = 55
E	5xH = 1500	5x(H+ (stream 400/32=13) + ndist[j] = 1) = 1,070
F	5 (add = 1) = 5	5 (ndist[i] = 1) = 5
G	5x2x(+,*,-,-,*,/)=60	5x2x(kown_points_x,y[j], neighbor[j],ndist[j] = 12) = 120
non-loop code after G	2 (-)	40
A	61,767	44,290

6. Design a pipelined accelerator for `distance()` that can perform distance computations at the rate `known_points[i]` data can be streamed to it at the maximum streaming rate (32 GB/s). Assume the `source_vector` input to `distance()` (which stays the same throughout the B loop) can be preloaded. For the appropriate consuming processor, assume `pdist[i]` outputs can be written into the associated small memory fast enough to keep up with streaming inputs and your designed accelrator.

   Hint: How many elements of `known_points[i]` can be delivered to the accelerator per cycle?
   How many subtracts does the accelerator need to perform to keep up with this rate of inputs?

   32 GB/s = 32B/ns = 32 B per cycle; 4B per element $\rightarrow$ 8 elements per cycles $\rightarrow$ 8 substracts per cycle



(This page intentionally left mostly blank for answers.)

7. Identify concurrency opportunties between loops.

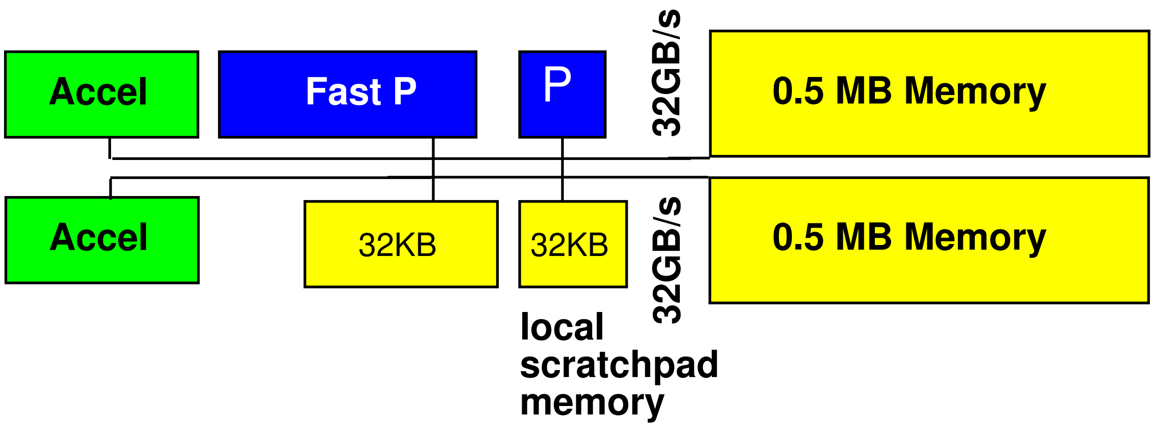
    (a) which loops can run concurrently, as separate processes, to increase the **throughput** for loop A

    (b) which loops can run concurrently, as separate processes, to reduce the **latency** for loop A (from read_sources() to new values for x and y)

	Throughput?	Latency?	Why?
B + C	Y	Y	Can stream pdist[i] to run loops concurrently.
C + D	Y	N	Must wait for C loop to complete to continue with D
D + E	Y	Y	can stream neighbor[j] from D to E and run concurrently
E + F	Y	Y	Can stream ndist[i] to run loops concurrently.
F + G	Y	N	Need totdist from F before can start G

(This page intentionally left mostly blank.)

8. Map the Loop A computation to a system composed of one simple processor (1 GHz as previously outlined), one fast processor (2 GHz, with everything running 2× as fast except data transfer from main memory), and two accelerators (Problem ??). Assume you have separate paths to the two large memory banks for each accelerator so they can both simultaneously stream at full rate.



Describe how you would map the computation onto these heterogeneous computing resources. Describe how you would use the scratchpad memories as necessary beyond what you've already answered in Problem ??. Estimate the performance your mapping achieves in cycles per loop A iteration.

Run as pipeline with:
1st stage – B running data parallel on 2 accelerators – each accelerator runs one distance in $\frac{400}{32} = 13$ cycles; 100 distances per accelerator – 1300 cycles
2nd stage – C, D, E running on fast processor (compute: 1700/2)+(memory: (C: 200/2) + (D: 5/2+50)+ (E: 5*(200/2+13+1/2)))=1570
3rd stage – F, G, rest running on slow processor – 67+165=232
These run concurrently (see previous problem), so cycles per A loop is max of these times or 1570 cycles.

(This page intentionally left mostly blank for answers.)

Code of Academic Integrity

Since the University is an academic community, its fundamental purpose is the pursuit of knowledge. Essential to the success of this educational mission is a commitment to the principles of academic integrity. Every member of the University community is responsible for upholding the highest standards of honesty at all times. Students, as members of the community, are also responsible for adhering to the principles and spirit of the following Code of Academic Integrity.*

Academic Dishonesty Definitions

Activities that have the effect or intention of interfering with education, pursuit of knowledge, or fair evaluation of a student's performance are prohibited. Examples of such activities include but are not limited to the following definitions:

A. Cheating Using or attempting to use unauthorized assistance, material, or study aids in examinations or other academic work or preventing, or attempting to prevent, another from using authorized assistance, material, or study aids. Example: using a cheat sheet in a quiz or exam, altering a graded exam and resubmitting it for a better grade, etc.

B. Plagiarism Using the ideas, data, or language of another without specific or proper acknowledgment. Example: copying another person's paper, article, or computer work and submitting it for an assignment, cloning someone else's ideas without attribution, failing to use quotation marks where appropriate, etc.

C. Fabrication Submitting contrived or altered information in any academic exercise. Example: making up data for an experiment, fudging data, citing nonexistent articles, contriving sources, etc.

D. Multiple Submissions Multiple submissions: submitting, without prior permission, any work submitted to fulfill another academic requirement.

E. Misrepresentation of academic records Misrepresentation of academic records: misrepresenting or tampering with or attempting to tamper with any portion of a student's transcripts or academic record, either before or after coming to the University of Pennsylvania. Example: forging a change of grade slip, tampering with computer records, falsifying academic information on one's resume, etc.

F. Facilitating Academic Dishonesty Knowingly helping or attempting to help another violate any provision of the Code. Example: working together on a take-home exam, etc.

G. Unfair Advantage Attempting to gain unauthorized advantage over fellow students in an academic exercise. Example: gaining or providing unauthorized access to examination materials, obstructing or interfering with another student's efforts in an academic exercise, lying about a need for an extension for an exam or paper, continuing to write even when time is up during an exam, destroying or keeping library materials for one's own use., etc.

* If a student is unsure whether his action(s) constitute a violation of the Code of Academic Integrity, then it is that student's responsibility to consult with the instructor to clarify any ambiguities.