Midterm

Started: Oct 7 at 6:45pm

Quiz Instructions

Regulations: https://www.seas.upenn.edu/~ese532/fall2020/midterm_details.pdf

Question 1

1 pts

I certify that I have complied with the University of Pennsylvania's Code of Academic Integrity and the exam regulations **https://www.seas.upenn.edu/~ese532/fall2020/midterm_details.pdf** (https://www.seas.upenn.edu/~ese532/fall2020/midterm_details.pdf) in completing this exam.

● True

○ False

Consider the following code in answering the questions on this exam.

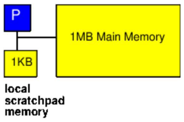
```
#include <stdint.h>
#define NUM_POINTS 1000
#define LOG2_NUM_POINTS 10
#define MAX_AREA (((uint64_t)1<<63)-1)
#define MAX_TIME (((uint64_t)1<<63)-1)
uint64_t min(uint64_t a, uint64_t b); // assume single instruction
uint64_t max(uint64_t a, uint64_t b); // assume single instruction
extern int **tp1set, **tp2set, **ap1set; // can hold negative numbers
extern int *dom;
extern uint64_t *ma, *mt;
uint64_t area_param(int arg, int num, int *a)
{
 uint64_t res=0;
 for (int i=0;i<num;i++) // loop F
  {
   int b=(arg & 0x01);
   arg=arg>>1;
   res+=b*a[i];
  }
 return(res);
}

uint64_t time_param(int arg, int num, int *t1, int *t2)
{
 uint64_t res=0;
 for (int i=0;i<num;i++) // loop G
  {
   int b=(arg & 0x01);
   arg=arg>>1;
   int tmp=(b*t1[i]+res);
   int t2i=t2[i];
   if (tmp==t2i)
      res=res+1;
   else
     res=max(t2i,res);
  }
 return(res);
}
void opt (int *tp1, int *tp2, int *ap1,
 int *non_dom_count_ptr, uint64_t *min_area_ptr, uint64_t *min_time_ptr)
{
 uint64_t a[NUM_POINTS];
 uint64_t t[NUM_POINTS];
 uint16_t dom[NUM_POINTS];

 uint64_t min_area=MAX_AREA;
 uint64_t min_time=MAX_TIME;
 uint64_t non_dom_count=0;
 for (int i=0;i<NUM_POINTS;i++) // loop A
  {
   a[i]=area_param(i,LOG2_NUM_POINTS,ap1);
   t[i]=time_param(i,LOG2_NUM_POINTS,tp1,tp2);
   dom[i]=0;
  }
 for (int i=0;i<NUM_POINTS;i++) // loop B
  {
   min_area=min(a[i],min_area);
   min_time=min(t[i],min_area);
  }
 for (int i=0;i<NUM_POINTS;i++) // loop C
  for (int j=0;j<NUM_POINTS;j++) // loop D
   {
     if ((i!=j) && (a[j]<=a[i]) && (t[j]<=t[i])) dom[i]++;
   }
 for (int i=0;i<NUM_POINTS;i++) // loop E
   {
    if (dom[i]==0) non_dom_count++;
   }
 *non_dom_count_ptr=non_dom_count;
 *min_area_ptr=min_area;
 *min_time_ptr=min_time;
}

void multi_opt(int num)
{
 for (int i=0;i<num;i++) // loop H
  {
   opt(*tp1set,*tp2set,*ap1set,dom,ma,mt);
   dom++;
   ma++;
   tp1set++;
   tp2set++;
   ap1set++;
  }
}
```

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, logical operations (&&, ||), min, max, 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 listed compute and memory operations. (Some consequences: You may ignore loop and conditional overheads in processor runtime estimates; you may ignore computations in array indicies.)
- Baseline processor can execute one multiply, compare, or add per cycle and runs at 1 GHz.
- Reads from and writes to the 1 MB main memory issue in one cycle, but require 5 cycles of latency (including issue) to get a result; memory can supply one read or write each cycle.
- Reads from and writes to the 1 KB scratchpad memory take 1 cycle.
- By default, all arrays live in the main memory and all array references are to main memory.
- Assume non-array variables live in registers.
- Assume all additions are associative. Max and min are associative.
- Assume comparisons, adds, min, max, 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.
- A lookup in a small memory (1KB or small) can complete in 1ns.

Question 2

10 pts

Estimate the time in cycles to run opt() sequentially on the single-processor baseline system described above.   Show your work for partial credit consideration.

HTML Editor

12pt Paragraph

0 words

Question 3

5 pts

For the single-processor implementation, identify the bottleneck top-level loop.

○ Loop A

○ Loop B

○ Loop C

○ Loop E

Question 4

5 pts

What is the maximum Amdahl's Law speedup if we only accelerate the loop identified in Question 3. (show work for partial credit consideration)

HTML Editor

12pt Paragraph

0 words

Question 5

10 pts

Consider the coarse-grained dataflow graph for the top-level loops (A, B, C, E). What precedence constraints exist among these loops (what producer->consumer relationships exist).

□ LoopA-->LoopB

☐ LoopB-->LoopC

☐ LoopC-->Loop E

☐ LoopE-->LoopA

☐ LoopA-->LoopC

☐ LoopA-->LoopE

☐ LoopB-->LoopE

☐ LoopC-->LoopB

☐ LoopC-->LoopA

☐ LoopE-->LoopB

☐ LoopE-->LoopC

☐ LoopB-->LoopA

Question 6 — 7 pts

Identify the loops that are data parallel.

☐ Loop A

☐ Loop B

☐ Loop C

☐ Loop D

☐ Loop E

☐ Loop F

☐ Loop G

Question 7 — 7 pts

Explain why each loop above is data parallel or not.

HTML Editor

12pt Paragraph

0 words

Question 8 — 10 pts

What is the Critical Path Latency Bound for the opt() function?

HTML Editor

12pt Paragraph

0 words

Question 9 — 5 pts

Accelerate the code on the baseline system by using the scratchpad memory.

Show your revisions to the code.   You only need to show the code you revised.

Hint: Since we're only asking for performance numbers to two significant figures, don't waste time on speedups that will have an impact of less than 1% .

HTML Editor

12pt Paragraph

0 words

Question 10

5 pts

Estimate the runtime and speedup for your revised code in Question 9.

HTML Editor

12pt Paragraph

0 words

Question 11

5 pts

When pipelined, what is the minimum clock cycle time achievable for loop G in time_param()?

[Do not unroll the loop. Think about pipelining the loop body to start one new iteration of the body on each clock cycle.]

[Assume you can have small local memory banks to hold t1 and t2.]

HTML Editor

12pt Paragraph

0 words

Question 12

10 pts

Design a pipeline for loop G in the time_param() calculation that achieves the minimum cycle time (in ns) [as identified in the previous question].

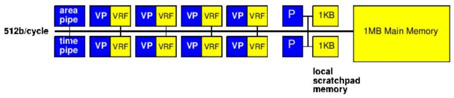
[Continue to assume you can have small local memory banks to hold t1 and t2.]

Upload Choose a File

Question 13

10 pts

Describe how to map the multi_opt() computation to the following heterogeneous system to maximize the throughput of opt() calculations.



- 1 instance of 1 GHz, II=1 area_param (loop F) calculation pipeline
  (assume have local memory banks for ap1 that can be used for each function invocation)
- 1 instance of time_param (loop G) calculation pipeline (as you designed above)
  (assume have local memory banks for tp1, tp2 that can be used for each function invocation)
- 2 single-issues, baseline processors (P), each running at 1 GHz.
- 8 Vector Processors (VP) with 8, 64b-wide vector lanes, each running at 1 GHz.
  (for the loops that are data-parallel, you may assume computation achieves the resource bound)
- There is a shared, 512b wide path to main memory available to the hardware pipelines and the Vector Processors. It can transfer one contiguous blocks of 512b into a Vector Register File (VRF) or hardware pipe each 1 ns cycle, but it takes 5 cycles of latency before a fetched value is available in the VRF or pipe. The single-issue, baseline processors can only move 64b from the main memory in cycle.

HTML Editor

12pt Paragraph

0 words

Question 14

5 pts

Estimate the throughput of your mapped multi_opt() design in cycles (1 ns cycles) per opt() calculation completion.

HTML Editor

12pt Paragraph

0 words

Question 15

5 pts

Estimate the latency of the opt() calculation for your mapped multi_opt() design in cycles (1 ns cycles) to compute each opt() result from when opt() first looks at its ap1, tp1, tp2 inputs.

HTML Editor

12pt Paragraph

0 words

Quiz saved at 6:46pm Submit Quiz