ECE 451-566: Introduction to Parallel and Distributed Computing, Fall 2020

Instructor: Maria Striki

Homework 1 (4 + 8 + 8 + 4 + 8 + 6 + 12 = 50 points)

Issued Date: Sat 09/19/20 Due Date: Wed 09/30/20 at 8.00pm

One submission per group (write ALL names on your Group solutions deliverable)

Problem 1: (4 points)

Does the addition of cache and virtual memory to a von Neumann system change its designation as an SISD (Single Instruction Single Data) system? What about the addition of pipelining? Multiple issue? Hardware multithreading? A von Neumann system consists of a central processor with an arithmetic/logic unit and a control unit, a memory, mass storage, input and output.

Solution:

Problem 2: (8 points)

How can we (if we can!) modify the following loops so that they can be maximally parallelized. As a first step identify and clearly describe what sort of dependencies exist in every loop (instruction per instruction and loop iteration per loop iteration). Clearly justify your answer whether the loop is parallelizable or not and if so how.

Loop 1:

```
for (i=0;i<100;i++) {
A[i] = A[i] * B[i]; /* S1 */
B[i] = A[i] + c; /* S2 */
A[i] = C[i] * c; /* S3 */
C[i] = D[i] * A[i]; /* S4 */
```

Loop 2:

```
for (i=0;i < 100;i++) {
A[i] = A[i] + B[i]; /* S1 */
B[i+1] = C[i] + D[i]; /* S2 */
}
```

Solution:

Problem 3: **(8 points)**

Part 1: (3pts + 3 pts + 2 pts) Can the loop below can be executed safely in a data-parallel model language using loop parallelism (i.e. fictitious "forall" library function? Yes/No, Why)?

C pseudo code :

```
for ( I = 1; I < 1000000 +1 ; I++ ) {
       Y[I] = X[I-1] + Y[I] + Z[I+1] ;
}
```

Part 2: Can the loop below can be executed safely in a data-parallel model language using loop parallelism (i.e. fictitious "forall" library function? If so, under which circumstances)?

```
nsize = sizeof(a);
#pragma concurrent
for (i=0; i< nsize; i++)
      A[i] = a[i+m];
```

Part 3: Can the loop below can be executed safely in a data-parallel model language using loop parallelism (i.e. fictitious "forall" library function? If so, under which circumstances?

```
for (i=0; i< nsize; i++) printf ("element A[%d] = %f\n",i,a[i]);
```

Solution:

Problem 4: **(4 points)**

Let's look into a single-core single-processor system with CPU speed of 4GHz. Attached to the chip of the processor there is only one level cache (L1) of size 48KB. The main memory connects to the CPU via a interconnect bus of finite speed. The DRAM (main memory) has size of 418 MB. The latency to L1 is 5 cycles and the latency to main memory is 150 cycles. In each memory cycle, the processor fetches 8 words (cache line has size of 8 words). What is the peak achievable performance of the dot product of two vectors?

Note: Peak achievable performance is the maximum performance that can be achieved during execution of the above program: i.e., how many operations or (FLOPS) in how many seconds or clock cycles?

```
for (i=0; i<dim; i++)
              dot_prod += a[i] * b[i];
```

Solution:

Problem 5: **(8 points)**

Assume the following instruction stream or kernel:

```
for (int x=0; x<dimX-1; x++) {
for (int y=0; y<dimY-1; y++) {
for (int z=0; z<dimZ-1; z++) {
     int index = x*dimY*dimZ + y*dimZ + z;
     if (y>0 && x >0) {
          solid = idx[index];
          dH1 = (Hz[index] – Hz[index-incY])/dy[y];
          dH2 = (Hy[index] – Hy[index-incZ])/dz[z];
          Ex[index] = Ca[solid]*Ex[index]+Cb[solid]*(dH2-dH1);
}}}}
```

It is given that dH1, dH2, Hy, Hz, dy, dz, Ca, Cb, Ex are single-precision floating-point arrays and idx is an unsigned int array.

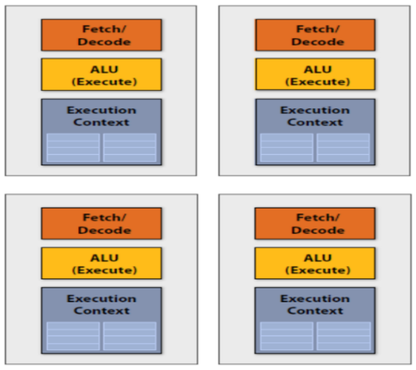
Part 1: What is the arithmetic intensity (how many operations performed in a certain number of clock cycles) of this instruction stream/kernel? **(2 points)**

Part 2: Is this instruction stream/kernel amenable to vector or SIMD execution? Why or why not? **(4 points)**

Part 3: Assume this kernel is to be executed on a processor that has 20 GB/sec of memory bandwidth. Will this kernel be memory bound or compute bound? **(2 points)**

Solution:

Problem 6: (6 points)



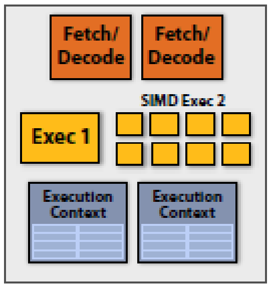
Picture 2

Part 1 (4 points): In the processor configuration of picture 2, in which component(s) of the core should I add more units to achieve maximum efficiency (better parallelism with lower cost) and what is an upper bound for each case when:

a) I have a scientific application which focuses on heavy uniform data calculations? **(2 points)**
b) I am running multiple diverse programs from various users **(2 points)**

Describe exactly the implications on all other modules when changing the component of preference, and describe in detail what sort of problems correspond to each case. You may propose more than one changes for each case.

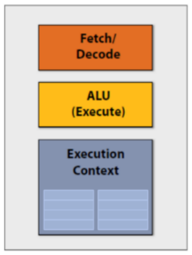
Solution:



Picture 3

Part 2 (2 points): Now let's assume that pic 3 is part of one core alone. And assume that our processor is quad-core. Are we allowed to use such as system within a core? Yes/no, why? Let's further assume (perhaps by mistake…) that we are allowed to use this architecture within a single core. Then, under what category can we classify our quad-core system (SISD, SIMD, MIMD, Superscalar)? What type of instructions should I execute in order to achieve *maximal utilization of resources* and when and how is this achieved?

Problem 7: (12 points)



Pic 1: a simple single-core processor

Part 1 (4 points): Draw pictures of the following multi-core machines:

a) 2 core, 6 SMT, 4-wide SIMD capability, **(2 points)**,

b) 2 core, 2 interleaved/temporal Multithreading, 6-wide SIMD capability, **(2 points)**, and

In each picture, you need to clearly illustrate the type and numbers of Functional Units (FUs), the Arithmetic Logical Units (ALUs), and Execution Contexts (ECs).

Part 2 (8 points): Based on the pictures you have illustrated above, answer the following questions, for each of your illustrations:

1) **(2 pts)** How many independent instructions can run simultaneously and how many concurrently?
2) **(2 pts)** Which scheme offers the maximum and which the minimum memory latency reduction? Why?
3) **(2 pts)** How many pieces of independent data are needed to run the chip with maximum latency hiding ability? Which scheme offers the max latency hiding ability and why?
4) **(2 pts)** Which of the above processor architectures would you use to run a heavy computation-bound SIMD application? And which one would you use for a distributed multi-user, multi-purpose system?

Solution: