ECE4700J Computer Architecture

Summer 2022

HW #5

Due: **2:59am (Beijing Time) July 20th, 2022**

*Please submit a single **PDF** file on Canvas*

Q1 (10%): Based on what you have learned, please list at least three differences between vector processors vs. SIMD extensions (such as Multimedia Extensions).

Sample Solutions:

SIMD extensions are extra instructions (MMX, SSE and AVX) that were added to the x86 architecture to support vector like operations. Like vector instructions, a SIMD instruction specifies the same operation on vectors of data. Unlike vector instructions, SIMD instructions tend to specify fewer operands and hence use much smaller register files.

- Multimedia SIMD extensions fix the number of data operands in the opcode. Vector architectures have a vector length register that specifies the number of operands for the current operation. These variable-length vector registers easily accommodate programs that naturally have shorter vectors than the maximum size the architecture supports. Moreover, vector architectures have an implicit maximum vector length in the architecture, which combined with the vector length register avoids the use of many opcodes.
- Multimedia SIMD does not offer the more sophisticated addressing modes of vector architectures, namely strided accesses and gather-scatter accesses. These features increase the number of programs that a vector compiler can successfully vectorize.
- Multimedia SIMD usually does not offer the mask registers to support conditional execution of elements as in vector architectures.
- …

Source: http://thebeardsage.com/simd-multimedia-extension/

Q2 (60%): Consider the following three CPU organizations:

CPU SS: A two-core superscalar microprocessor that provides out-of-order issue capabilities on two function units (FUs). Only a single thread can run on each core at a time.

CPU MT: A fine-grained multithreaded processor that allows instructions from two threads to be run concurrently (i.e., there are two functional units), though only instructions from a single thread can be issued on any cycle.

CPU SMT: An SMT processor that allows instructions from two threads to be run concurrently (i.e., there are two functional units), and instructions from either or both threads can be issued to run on any cycle.

Assume we have two threads X and Y to run on these CPUs that include the following operations:

Thread X	Thread Y
A1 – takes three cycles to execute	B1 – take two cycles to execute
A2 – no dependences	B2 – conflicts for a functional unit with B1
A3 – conflicts for a functional unit with A1	B3 – depends on the result of B2
A4 – depends on the result of A3	B4 – no dependences and takes two cycles to execute

Assume all instructions take a single cycle to execute unless noted otherwise or they encounter a hazard.

1) Assume that you have one SS CPU. How many cycles will it take to execute these two threads? How many issue slots are wasted due to hazards?
2) Now assume you have two SS CPUs. How many cycles will it take to execute these two threads? How many issue slots are wasted due to hazards?
3) Assume that you have one MT CPU. How many cycles will it take to execute these two threads? How many issue slots are wasted due to hazards?
4) Assume you have one SMT CPU. How many cycles will it take to execute the two threads? How many issue slots are wasted due to hazards?

Sample Solutions:

(Answers can vary slightly)

1)

Core 1	Core 2
A3	B1, B4
A1, A2	B1, B4
A1, A4	B2
A1	B3

2) The same as 1).

3)

FU1	FU2
A1	A2
A1	
A1	
B1	B2
B1	
A3	
A4	
B4	
B4	

4)

FU1	FU2
A1	B1
A1	B1
A1	B2
A2	B3
A3	B4
A4	

Q3 (30%): This problem concerns MSI, an invalidation based snooping cache coherence protocol, for bus-based shared-memory multiprocessors with a single level of cache per processor. A block starting at address Addr can be in one of the following states in cache C:

- **Modified**: The block is present only in cache C and the data in the cache is dirty or modified (i.e., it reflects a more recent version than the copy in memory).

- **Shared**: The block is present in cache C and possibly present in other caches.
- **Invalid**: The block is not valid in cache C (space for the block may or may not be currently allocated in this cache).

Consider the following sequence of operations by two processors for a block that starts at address B. Determine the state of that block in the caches of both the processors after each operation in the sequence for the MSI protocol. Both caches are initially empty and all lines are in the I state. The table below is provided to help organize your answer.

No.	Operation	MSI	
		P1	P2
1	P1 reads B	S	I
2	P1 writes B	M	I
3	P2 writes B	I	M
4	P1 reads B	S	S
5	P1 writes B	M	I
6	P2 reads B	S	S
7	P2 writes B	I	M