1. Potpourri [80 points]

(a) For each of the following, circle if the concept is a part of the ISA versus the microarchitecture (circle only one):

Number of threads in fine-grained multithreading
Circle one: ISA Microarchitecture

Number of DRAM banks
Circle one: ISA $Microarchitecture$

Vectored interrupts
Circle one: ISA Microarchitecture

Number of entries in reservation stations
Circle one: ISA $Microarchitecture$

Number of entries in the reorder buffer
Circle one: ISA $Microarchitecture$

Number of entries in the architectural register file
Circle one: ISA Microarchitecture

Number of entries in the physical register file
Circle one: ISA $Microarchitecture$

Number of sets in the L3 cache
Circle one: ISA $Microarchitecture$

The page table base register of the executing process
Circle one: ISA Microarchitecture

(b) Why does ISA change more slowly than microarchitecture?

> ISA is exposed to the programmer. Adopting a new or modified ISA requires changes to software and compilers, whereas adopting a new microarchitecture requires no such changes.

(c) A program is written in C. We execute this program on two different computers:

Computer A: has a processor that implements the x86 ISA and has 3 GHz clock frequency
Computer B: has a processor that implements the x86 ISA and has 3 GHz clock frequency

When we execute this program and measure its cycles per instruction (CPI) in x86 instructions, we find the following result:

On Computer A: CPI is equal to 10
On Computer B: CPI is equal to 8

What can you say about on which computer (A or B) this program runs faster?

We don't know.

Explain and show all your work below:

Because we don't know how many instructions are actually executed for the program on either machine, we cannot conclude which computer runs faster. Although B has lower CPI, but it might be executing 2 times more instructions than A due to a less optimized compiler.

(d) You are designing an ISA that uses delayed branch instructions. You are trying to decide how many instructions to place into the branch delay slot. How many branch delay slots would you need for the following different implementations? Explain your reasoning briefly.

An in-order processor where conditional branches resolve during the 4th stage:

3

An out-of-order processor with 64 unified reservation station entries where conditional branches resolve during the 2nd cycle of branch execution. The processor has 15 pipeline stages until the start of the execution stages.

We don't know.

(e) What three key pieces of information does the compiler not know when performing instruction scheduling?

Memory addresses

Cache hit/miss status

Branch direction

(f) In class, we discussed the concept of traces and trace scheduling.

What is a trace?

A frequently executed sequence of basic blocks.

Now suppose we make each trace atomic, as we also discussed in class. What is the benefit of making each trace atomic? Explain.

Enables more compiler optimizations. The compiler can freely reorder instructions within the atomic trace subject only to true dependencies, without requiring any fix-up code.

What is the disadvantage?

Wasted work when the atomic trace that is executed is not for the control flow path that is supposed to be executed.

(g) Assume we have an ISA with virtual memory. It is byte-addressable and its address space is 64 bits. The physical page size is 8KB. The size of the physical memory we use in a computer that implements the ISA is 1 TB (2^{40} bytes).

Assume the demand paging system we would like to design for this uses the perfect LRU algorithm to decide what to evict on a page fault. What is the minimum number of bits that the operating system needs to keep to implement this perfect LRU algorithm? Show your work.

$\lceil \log_2 2^{27}! \rceil$