Chapter 4

1. Is it reasonable to build a scalar pipeline that supports out-of-order execution? If so, describe a code execution scenario where such a pipeline would perform better than a conventional in-order scalar pipeline.

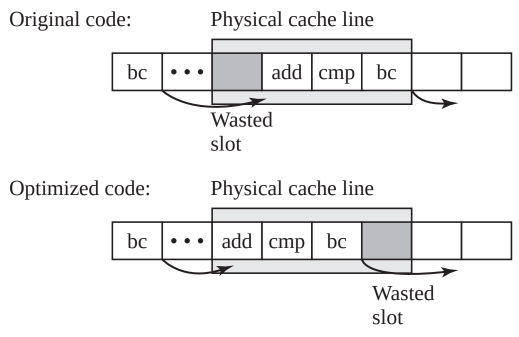
   Yes. An example scenario is when a load instruction stalls on a cache miss. Subsequent independent instructions can still make progress if the pipeline supports out-of-order execution.

2. Superscalar pipelines require replication of pipeline resources across each parallel pipeline, naively including the replication of cache ports. In practice, however, a two-wide superscalar pipeline may have two data cache ports but only a single instruction cache port. Explain why this is possible, but also discuss why a single instruction cache port can perform worse than two (replicated) instruction cache ports.

   The spatial locality in the instruction reference stream is quite high. Hence, a single port that is twice as wide can usually satisfy both pipelines by fetching two adjacent instructions. The exception to this condition is when the first instruction fetched in the cycle is a taken branch. Whenever this is the case, the spatially adjacent instruction is not useful, since it lies on the fall-through path. In this case, assuming the fetch stage recognized that a taken branch was being fetched for the first pipeline, it could use the second instruction cache port to fetch an instruction from the branch target.

3. Section 4.3.1 suggests that a compiler can generate object code where branch targets are aligned at the beginning of physical cache lines to increase the likelihood of fetching multiple instructions from the branch target in a single cycle. However, given a fixed number of instructions between taken branches, this approach may simply shift the unused fetch slots from before the branch target to after the branch that terminates sequential fetch at the target. For example, moving the code at label0 so it aligns with a physical cache line will not improve fetch efficiency, since the wasted fetch slot shifts from the beginning of the physical line to the end. .

```
    bc cond, label0
    • • •
label0:
    add r1, r2, r3
    cmp cond, r1, r5
    be cond, label1
```



   Discuss the relationship between fetch block size and the dynamic distance between taken branches. Describe how one affects the other, describe how important is branch target alignment for small vs. large fetch blocks and short vs. long dynamic distance, and describe how well static compiler-based target alignment might work in all cases.

The essay should point out the following: As fetch block size increases, the probability of encountering a taken branch within the block increases. The greater the dynamic distance between taken branches is, the better a simple fetch mechanism will work, since taken branches cause discontinues in the fetched instruction stream. For small fetch blocks, aligning branch targets is important, since relatively more fetch slots are lost even with minor misalignment. However, for large fetch blocks, major misalignment can also cause severe fetch bandwidth loss. The bottom line is that target alignment will only be beneficial if the target block that is being aligned contains enough instructions to satisfy fetch bandwidth. If the target block itself is short, whether it is aligned or not will not matter. Hence, a compiler heuristic that is sensitive to target block size could improve fetch bandwidth by only aligning targets in those cases where the target block exceeds some threshold.

4. The auto-realigning instruction fetch hardware shown in Figure 4-13 still fails to achieve full-width fetch bandwidth (i.e., four instructions per cycle). Describe a more aggressive organization that is always able to fetch 4 instructions per cycle. Comment on the additional hardware such an organization implies.

   An even/odd interleaved cache would work, where each bank is able to fetch four instructions per cycle. An 8:4 mux connects the output of the two banks and can always deliver four instructions (assuming both banks hit).

5. One idea to eliminate the branch misprediction penalty is to build a machine that executes both paths of a branch. In a 2-3 paragraph essay, explain why this may or may not be a good idea.
6. Section 4.3.2 discusses adding predecode bits to the instruction cache to simplify the task of decoding instructions after they have been fetched. A logical extension of predecode bits is to simply store the instructions in decoded form in a decoded instruction cache; this is particularly attractive for processors like the Pentium Pro that dynamically translate fetched instructions into a sequence of simpler RISC-like instructions for the core to execute. Identify and describe at least one factor that complicates the building of decoded instruction caches for processors that translate from a complex instruction set to a simpler RISC-like instruction set.

   Possible factors include: unequal expansion ratios (some CISC sequences will expand to more instructions than others); self-modifying code (where cached sequences must be invalidated); alignment issues with a fixed cache line length in the decoded cache.

7. What is the most important advantage of a centralized reservation station over distributed reservation stations?

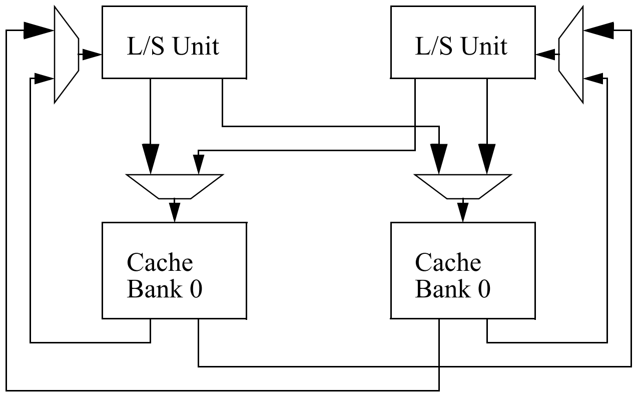
   Better utilization (which is always true for a shared resource).

8. In an in-order pipelined processor, pipeline latches are used to hold result operands from the time an execution unit computes them until they are written back to the register file during the write-back stage. In an out-of-order processor, rename registers are used for the same purpose. Given a four-wide out-of-order processor TYP pipeline, compute the minimum number of rename registers needed to prevent rename register starvation from limiting concurrency. What happens to this number if frequency demands force a designer to add five extra pipeline stages between dispatch and execute, and five more stages between execute and retire/writeback?

For maximum throughput, each pipeline stage will contain four inflight instructions. Since registers are allocated at decode, and freed at retire, each instruction holds a rename register for a minimum of five cycles. Hence 4x5 = 20 rename registers are needed at minimum, assuming no data dependences or cache misses that would cause instructions to stall. Adding five extra stages would increase the minimum to 40 registers. Of course, the students should understand that this is a minimum that assumes no data dependences or cache misses. At the same time, it also assumes throughput of 4 IPC. Since few processors achieve 4 IPC on real programs due to data dependences, control dependences, and cache misses, this “minimum” may in fact be sufficient. The only reliable way to determine the right number of rename registers is a sensitivity study using detailed simulation of a range of rename registers to find the knee in the performance curve.

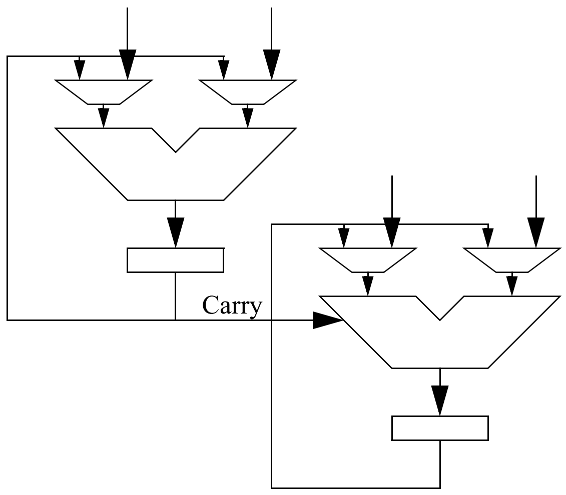
9. A banked or interleaved cache can be an effective approach for allowing multiple loads or stores to be performed in one cycle. Sketch out the data flow for a two-way interleaved data cache attached to two load/store units. Now sketch the data flow for an eight-way interleaved data cache attached to four load/store units. Comment on how well interleaving scales or does not scale.

   The figure below shows the two-way interleaved case. The student should point out that a crossbar interconnect between the load/store units and the cache ports is required, and that crossbars, in general, do not scale well. In the figure, the crossbar is implemented with multiplexers.



10. The Pentium 4 processor operates its integer arithmetic units at double the nominal clock frequency of the rest of the processor. This is accomplished by pipelining the integer adder into two stages, computing the low-order 16 bits in the first cycle and the high-order 16 bits in the second cycle. Naively, this appears to increase ALU latency from one cycle to two cycles. However, assuming that two dependent instructions are both arithmetic instructions, it is possible to issue the second instruction in the cycle immediately following issue of the first instruction, since the

low-order bits of the second instruction are dependent only on the low-order bits of the first instruction. Sketch out a pipeline diagram of such an ALU along with the additional bypass paths needed to handle this optimized case.



11. Given the ALU configuration described in Problem 10, specify how many cycles a trailing dependent instruction of each of the following types must delay, following the issue of a leading arithmetic instruction: arithmetic, logical (and/or/xor), shift left, shift right.

    Assuming the leading instruction is arithmetic:

    - arithmetic: 0
    - logical: 0
    - shift left: 0
    - shift right: 1

12. Explain why the kind of bit-slice pipelining described in Problem 10 cannot be usefully employed to pipeline dependent floating-point arithmetic instructions.

    Because FP operations require a normalization step: for FP add or subtract, the significand may-need to be shifted by up to the difference in the exponents between the two operands. The shift may require bits from a higher-order bitslice, forcing the operation to stall. Furthermore, the shift amount will not be known until the exponent is known, and the exponent may change due to rounding or normalization, and cannot be determined until the high-order bitslices are done.

13. Assume a 4-wide superscalar processor that attempts to retire four instructions per cycle from the reorder buffer. Explain which data dependences need to be checked at this time, and sketch the dependence-checking hardware.

    In a machine with a separate architected register file and rename register file, the commit/retire hardware must check for write-after-write dependences to ensure that two instructions do not attempt to commit the same architected register value to the architected register file. Similarly, commit-stage updates to the rename table or map table need to be checked for conflicts. Specifically, there are WAW hazards within the four-instruction commit group if multiple instructions write the same logical register; there are WAW hazards from the commit group to the current dispatch group if a new instruction is renaming the same destination register; there are RAW hazards from the commit group to the current dispatch group if a new instruction needs to read a register that is being committed to the architected register file..

In a machine with a unified physical register file, there are no such hazards.

14. Four-wide superscalar processors rarely sustain throughput much greater than one instruction per cycle (IPC). Despite this fact, explain why 4-wide retirement is still useful in such a processor.

    Providing 4-wide retirement is beneficial whenever the processor's reorder buffer is completely full. In that case, the front-end bandwidth (dispatch bandwidth) is directly limited by the retirement bandwidth: if the processor can only retire 2 instructions per clock cycle, only 2 ROB slots are freed per cycle, and only 2 new instructions can enter the execution window. Providing 4-wide retirement prevents this from occurring.

15. Most general purpose instruction sets have recently added multimedia extensions to support vector-like operations over arrays of small data types. For example, Intel IA32 has added the MMX and SSE instruction set extensions for this purpose. A single multimedia instruction will load, say, eight 8-bit operands into a 64-bit register in parallel, while arithmetic instructions will perform the same operation on all eight operands in SIMD (single-instruction, multiple data) fashion. Describe the changes you would have to make to the fetch, decode, dispatch, issue, execute, and retire logic of a typical superscalar processor to accommodate these instructions.
    - Fetch: no change
    - Decode: modify to recognize new instructions
    - Dispatch: no change
    - Issue: no change
    - Execute: ALUs must support new operations
    - Retire: no change
16. The PowerPC instruction set provides support for a fused floating-point multiply-add operation that multiplies two of its input registers and adds the product to the third input register. Explain how the addition of such an instruction complicates the decode, dispatch, issue, and execute stages of a typical superscalar processor. What effect do you think these changes will have on the processor's cycle time?

    Most likely, the cycle time will be longer because of these changes:

    - Decode: rename logic must support 50% more source operands per cycle
    - Dispatch: no change
    - Issue: scheduling and wakeup logic must support 50% more operands (50% more comparators on each wakeup bus, increasing bus capacitance)
    - Execute: additional bypass paths and ALU logic to support fused multiply-add.

17. The semantics of the fused multiply-add instruction described in Problem 16 can be mimicked by issuing a separate floating-point add and floating-point multiply whenever such an instruction is decoded. In fact, the MIPS R10000 does just that; rather than supporting this instruction (which also exists in the MIPS instruction set) directly, the decoder simply inserts the add/multiply instruction pair into the execution window. Identify and discuss at least two reasons why this approach could reduce performance as measured in instructions per cycle (IPC).

    Such an approach will reduce effective decode, dispatch, issue, execute, and retire bandwidth, since each multiply-add will consume two slots instead of one in all these pipestages.

18. Does the ALU mix for the Motorola 88110 processor shown in Figure 3-7 agree with the IBM instruction mix provided in Section 2.2.4.3? If not, how would you change the ALU mix?

    Considering strictly integer instructions, and ignoring the 88110's graphics and floating point units: there are 3 integer units, 1 load/store unit, and 1 branch unit. Hence, 60% of execution bandwidth goes to integer instructions, 20% to loads and stores, and 20% to branches. The IBM mix reports 40% ALU instructions, 25% loads, 15% stores, and 20% branches. Hence, the most pressing mismatch occurs due to lack of execution units for loads and stores. The 88110 could either add a second load/store unit (best solution), or separate load and store functionality by providing one load unit and one store unit. This would result in a functional unit mix that fairly closely matches the IBM mix.

19. A: A mechanism that tracks out-of-order execution and maintains speculative machine state.
    Q: What is _the reorder buffer______________________ ?
20. A: It will significantly reduce the machine cycle time, but can increase the branch penalty.
    Q: What is _pipelining or deep pipelining______________ ?
21. A: Additional I-cache bits generated at cache refill time to ease the decoding/dispatching task.
    Q: What are _predecode bits________________________ ?
22. A: A program attribute that causes inefficiencies in a superscalar fetch unit.
    Q: What is _a taken branch_________________________ ?
23. A: The internal RISC-like instruction executed by the Pentium Pro (P6) microarchitecture.
    Q: What is __uop or micro-op______________________ ?
24. A: The logical pipeline stage that assigns an instruction to the appropriate execution unit.
    Q: What is __dispatch_____________________________ ?
25. A: An early processor design that incorporated ten diverse functional units.
    Q: What is __Motorola M88110____________________ ?
26. A: A new instruction that allows a scalar pipeline to achieve more than one floating-point operation per cycle.
    Q: What is __floating-point multiply-add or multiply-accumulate or MAC_ ?
27. A: An effective technique for allowing more than one memory operation to be performed per cycle.
    Q: What is _a dual-ported or multiported or multibanked cache____?
28. A: A useful architectural property that simplifies the task of writing low-level operating system code.

Q: What is _support for precise exceptions__ ?

29. A: The first research paper to describe runtime, hardware translation of one instruction set to another, simpler one.

    Q: What was __HPS or high-performance substrate by Patt, Hwu, and Shebanow__ ?

30. A: The first real processor to implement runtime, hardware translation of one instruction set to another, simpler one.

    Q: What was __the Intel Pentium Pro___________________ ?

31. A: This attribute of most RISC instruction sets substantially simplifies the task of decoding multiple instructions in parallel.

    Q: What was __fixed-length instructions, regular encoding, single destination register, no more than two source registers_ ?