CpSc-418 Final December 13, 2001
Solution Set

1. (18 points) This question is about the paper: "Exploiting Instruction Level Parallelism in Processors by Caching Scheduled Groups."

   (a) (5 points) Summarize the main idea of the paper in one or two sentences. In other words, how do the authors propose to get high performance?

   They propose translating instructions to groups of VLIW style instructions as the program executes. The VLIW engine is simple and fast. The translation process can span several basic blocks go create a group, thereby performing optimizations according to the executation paths that are actually encountered.

   Grading: A correct solution should mention translation to VLIW (or LIW, or "scheduled groups") and give at least one reason why this contributes to performance.

   (b) (5 points) Write two or three sentences to describe how register renaming is performed with DIF. Describe one advantage compared with the traditional (i.e. MIPS R10000 approach) (one sentence). Describe a situation where the traditional approach could work better (one sentence).

   For each logical register there are four physical registers. Dependencies within a group are handled by the translation, with separate instances of a register mapping to separate of the four physical registers. The renaming is relative to the mapping established by the previously executed group, and each group updates this base mapping on exit.

   This method is simpler than the MIPS approach because each logical register has a separate pool of physical registers and can be remapped independently. Most of the remapping complexity is handled when translating to VLIW groups. When a group is executed, the machine just adds the two bit specifier for which physical register to use to the two bit value in the base mapping.

   The traditional approach has more flexibility in mapping logical registers to physical ones. Thus, the DIF approach may run out of physical registers for one particular logical registers, while having lots of unused physical registers that are dedicated to other logical registers. For example, when running ahead with loop index calculations, DIF could run out of registers for the index variable, when the MIPS approach would not.

   Grading: A correct answer must mention that for each logical register, there is a dedicated set of physical registers. For advantages over the MIPS, the answer should point out that the DIF method is simpler and give at least one reason why. For advantages of the MIPS approach, the greater flexibility of the MIPS approach or some consequence/example of this flexibility should be described.

(c) (4 points) Give one reason from the paper that pure VLIW architectures perform poorly on commercial workloads. Does this consideration affect the IA-64?

The paper points out that commercial workloads make extensive use of dynamically linked libraries. A compiler cannot optimize across calls into such a library. To find good parallelism, VLIW compilers need to optimize over large blocks. This is prevented by dynamically linked code.

Another issue is code bloat. VLIW code is typically much larger than its non-VLIW counterpart. Commercial workloads tend to have poor locality in both instructions and data. Thus, a VLIW compiler must optimize (and therefore bloat) all or most of the executable code. DIF translates to VLIW on demand, and only expands the code that is currently in use.

The IA-64 suffers from both of these problems.

Grading: Either reason gets full credit.

(d) (4 points) The authors suggest that a DIF machine could achieve good performance without an L1 I-cache. How could this observation be used to simplify the design of the primary engine?

If the machine had no L1 cache, then the primary engine can only run at the L2 hit rate. While the L2 bandwidth may be high, the small basic blocks of typical code suggest that the primary engine will execute significantly less than one instruction per cycle. The observation that a DIF machine doesn't need an L1 cache to get good performance, means that the performance of the primary engine isn't critical. Thus, designers can choose simplicity over performance in the design of the primary engine. For example, it could take more than one cycle per instruction.

Grading: the answer should recognize that this observation means that the performance of the primary engine isn't critical and that this could allow simplifying the design.

2. (18 points) This question is about the paper: "A 0.18μm CMOS IA-32 Processor With a 4-GHz Integer Execution Unit."

(a) (6 points) Summarize the main idea of the paper in one or two sentences. In other words, how do the authors propose to get high performance?

The authors claim that, in general, increasing clock frequency increases processor performance even if it requires a deeper pipeline and therefore introduces more hazards and greater penalties. They state that performance grows slower than proportionally to clock frequency, but performance is monotonic with clock period.

Grading: Full credit for stating that they claim that higher frequency leads to higher performance even if it requires a deeper pipeline and the associated problems.

(b) (3 points) Based on the data given in the paper for Intel processors, by what factor can clock frequency be increased when the number of stages in the pipeline is doubled?

They claim a 1.5 or 1.6 $\times$ improvement in performance with doubling clock period based on the Pentium to Pentium-III and Pentium-III to Pentium-4 experience.

Grading: Full credit for an answer of 1.5 or 1.6. No explanation needed. Negative points for answers with more than 50 words.

(c) (3 points) A common instruction sequence is add $\rightarrow$ load $\rightarrow$ test $\rightarrow$ branch, where the add operation is address calculation for the load, and the test operation is an add or subtract to compare two values. Write one or two sentences to explain how the result of the add is used by the load. How many clock cycles at what frequency elapse from starting the add until starting the load?

The cache can use the result as it is generated by the (staggered) adder. The load can start one 4GHz clock cycle after the add.

Grading: Full credit for an answer of one clock cycle at 4 GHz (or 3 GHz, as they also describe a 1.5GHz version of the P-4).

(d) (3 points) For the same instruction sequence as part c, write two or three sentences to explain how the result of the load is used by the test. How many clock cycles at what frequency elapse from starting the load until starting the test if the load hits in the L1 data cache? What happens if the load misses in the L1 data cache?

The load to use delay is two cycles of the 4GHz clock (alternatively, one cycle of the 2GHz clock). If the load misses in the L1, then the micro-op for the test will proceed with the wrong value, and be aborted and re-issued when tag-comparison completes later.

Grading: Full credit requires noting the two cycle load-to-use delay, and noting the speculative use of the data before tag comparison completes.

(e) (3 points) For the same instruction sequence as part c, write two or three sentences to explain how the result of the test is used by the branch. How many clock cycles at what frequency elapse from starting the test until starting the branch? How does this impact overall processor performance?

The branch depends on flags that are set in the last cycle of the staggered add. Therefore, three cycles at 4 GHz elapse from the start of the test until the start of the branch. If the branch was mispredicted, this extra latency due to the staggered add contributes directly to the branch mispredict penalty.

Grading: Full credit requires noting that flags are set several cycles after the test operation starts executing, and noting the impact of this on the branch mispredict penalty.

3. (18 points) This question compares the papers: "Exploiting Instruction Level Parallelism in Processors by Caching Scheduled Groups" and "A 0.18μm CMOS IA-32 Processor With a 4-GHz Integer Execution Unit."

   (a) (10 points) What feature of the Pentium-4 most closely corresponds to a DIF cache? State two points of similarity between these two features, and state two differences. Your answer can be in point form; full sentences are not expected.

   • The Pentium-4 trace cache.
   + Both cache traces that span multiple basic blocks.
   + Both perform some amount of instruction decoding and reformatting before caching the data.
   – DIF does instruction scheduling and register renaming before the trace cache – the P4 does it afterwards.
   – The P4 always executes from the trace cache – DIF executes in the primary engine the first time a group is encountered and creates the cache entry in parallel with this execution.

   Grading: two points for identifying the trace cache as the corresponding feature. Two points for each of up to two similarities. Two points for each of up to two differences.

   (b) (4 points) Compare how DIF and the Pentium-4 handle complex instructions. A two or three sentence answer is adequate.

   DIF uses the primary engine.
   The P4 generates micro-op sequences from a micro-code ROM. Alternatively, some students may note that the P4 has three integer pipelines, two for simple operations, and another for complex ones.

   Grading: two points for the correct answer about DIF, and two points for a correct answer about the P4.

   (c) (4 points) Compare how DIF and the Pentium-4 provide precise exceptions. A two or three sentence answer is adequate.

   DIF exits the current group and resets register mappings back to the mappings from when the group was entered. It then re-executes the group one the primary engine where precise exceptions are relatively straightforward (i.e. the same basic mechanism as the MIPS R2000 from CpSc 318).
   The P4 uses a re-order buffer like the MIPS R10000. When an instruction raises an exception, it is held in the commit buffer until all previous (by program order) instructions have committed. All subsequent instructions (by program order) are aborted.

   Grading: two points for the correct answer about DIF, and two points for a correct answer about the P4.

4. (17 points)

(a) (5 points) How does the complexity of a super-scalar processor increase with the issue width? Give an asymptotic form (e.g. $O(\log n)$, $O(n)$, $O(n!)$, etc.). State two aspects of super-scalar design that lead to this complexity.

The necessary hardware resources grow as $O(w^2)$ where w is the issue width. This is because the registers written by the up to w instructions to be issued must have their identifiers compared with all registers accessed by the other w. This creates $O(w^2)$ interactions that must be checked in the same cycle. Similar complexity occurs for bypassing. The register file must have $O(w)$ read and write ports which requires $O(pw)$ height for the register file (where p is the number of physical registers in the register file), and $O(bw)$ width where b is the number bits in a register. The product is $O(pbw^2)$. In practice, p must grow with w as well, so one could make an argument for $O(w^3)$.

Grading: two points for the correct asymptotic bound. Three points for the examples: take of one point for a messed up example, two points for a completely wrong example, for a maximum of three points deducted.

(b) (3 points) How does the IA-64 architecture attempt to deal with the complexity from part (a)? Your answer should consist of one or two sentences.

The IA-64 is basically an VLIW machine. The compiler handles instruction scheduling and register renaming. The register file still needs lots of ports.

Grading: three points for noting that VLIW removes the need for hardware for instruction scheduling and/or register renaming (i.e. getting either one gets full credit).

(c) (3 points) How does the SMT architecture attempt to deal with the complexity from part (a)? Your answer should consist of one or two sentences.

SMT does little to address this. Because it exploits TLP, it can be a little less agressive about ILP and scale back somewhat on some of the complexity that goes into a high ILP machine.

Grading: equivalent to part (b).

(d) (3 points) How does the CMP architecture attempt to deal with the complexity from part (a)? Your answer should consist of one or two sentences.

CMP uses simple CPUs and avoids the overhead of high issue rates. The communication bottleneck moves to the interconnect between the processors and the L2 cache(s). This is only accessed for L1 misses (including coherency misses) rather than for every cache miss. While it's still $O(w^2)$, the constant factor is much smaller. In practice, CMP

complexity grows roughly with the number of processors.

Grading: three points for recognizing that CMP avoids the quadratic complexity of super-scalars by building simple cores.

(e) (3 points) How does the DIF architecture attempt to deal with the complexity from part (a)? Your answer should consist of one or two sentences.

DIF uses static scheduling like VLIW. The scheduling is done in the translation hardware between the primary engine and the DIF cache. While this must also check more interactions, it's not on the latency-critical path, and the extra time improves the constant factors dramatically.

Grading: three points for recognizing that DIF avoids the quadratic complexity of superscalars by using static scheduling.

5. (17 points) For each term below, give a one sentence definition.

(a) (3 points) ICOUNT.

A method for choosing which thread to fetch from in an SMT – the thread with the fewest instructions in flight is the one chosen for the next fetch. From "Exploiting Choice: Instruction Fetch and Issue on an Implementable Simultaneous Multithreading Processor."

Grading: full credit for correctly describing ICOUNT. I included the article reference for completeness. I don't expect such references in real solutions.

(b) (4 points) Predicated execution.

A method of selectively committing (or aborting) instructions after they issue based on flags set by other instructions.

(c) (3 points) Victim caching.

Storing the most recently evicted cache blocks in a small auxiliary cache to reduce conflict misses.

(d) (4 points) WAR hazard.

A Write-After-Read hazard where an instruction that writes register r can't issue until after the last instruction before the write in program order that reads register r has completed. WAR hazards can be eliminated by register renaming.

(e) (3 points) Zoning.

Partitioning the tracks of a disk so that outer tracks have more sectors than inner ones.

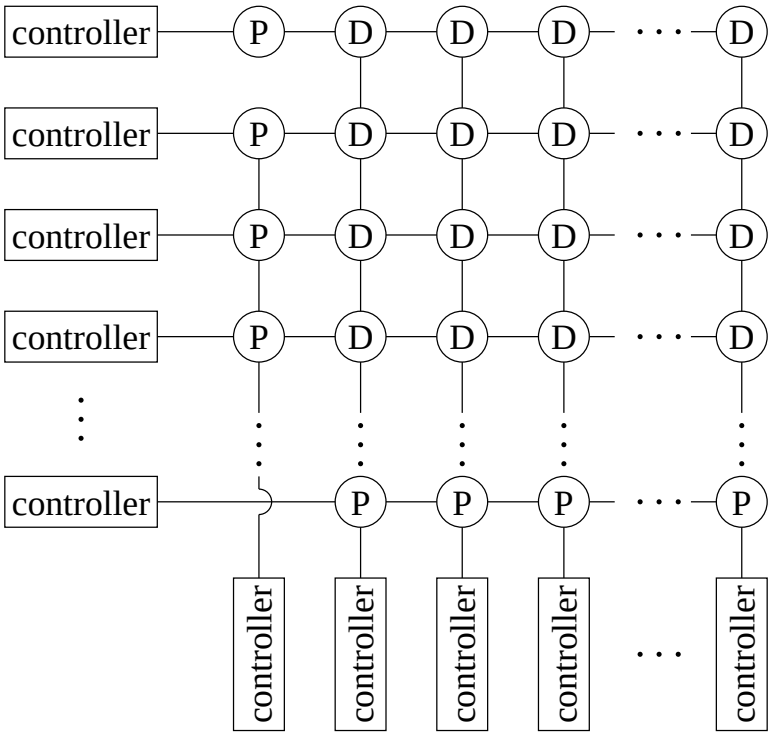

Figure 1: A Modified RAID Array

6. (17 points) This question pertains to the modified RAID array shown in figure 1. Each circle depicts a disk. Disks labeled with a "D" hold data, and those labeled with a "P" hold parity for their row or column. Each disk has interfaces to two controllers, one for its row, and one for its column.

   (a) (3 points) Explain why this array can survive any two disks failing without loss of data or availability.

   If the disks are in different rows and different columns, then their contents can be reconstructed by using either row or parity calculations.
   If both disks are in the same row, and neither is a parity disk, then their contents can be reconstructed using column parity calculations. Likewise, row parity can be used if they are both in the same column.
   If the two disks are in the same row and one is a parity disk, then the data disk is reconstructed using column parity, and then the parity disk is reconstructed using row parity. A similar approach works for a failed data and parity disk in the same column.

   Grading: One point for each of the three cases described above.

   (b) (3 points) Describe a three disk failure that results in loss of data.

   If a data disk and its corresponding row and column parity disks all fail, then none of them can be reconstructed.

   Grading: Three points for a correct answer. I don't believe any other three disk failure results in data-loss. For example, if a three data disks fail such that two are in the same row and two are in the same column:, for example:

   1 2
   3

   Then 2 can be reconstructed from column parity while 3 is reconstructed from row parity. After either 2 or 3 is reconstructed, then 1 can be reconstructed from row or column parity respectively. Take-off two points for an error of this form.

   (c) (3 points) Explain why this array can survive any controller failing without loss of data or availability.

   If a row controller fails, the data (or parity) is available from the column controllers. Likewise, if a column controller fails, then the data is available from the row controllers.

   Grading: Three points for a correct answer.

   (d) (8 points) Consider an array where each row has 20 data disks and one parity disk (except for the bottom row, which has 20 parity disks), and each column has 20 data disks and

one parity disk (except for the left column, which has 20 parity disks). Assume that disks have a mean-time-to-failure of 500,000 hours, and that disk failures are Poisson distributed. For simplicity, assume that disks are the only things that ever fail in this array (e.g. controllers never fail). Assume that once per day, any faulty disks are replaced.

What is the mean-time-to-data-loss for this array?

You may make and state reasonable assumptions to further simplify the problem and still receive full credit as long as your final answer is within a factor of two of the exact answer.

Assumption: independent failures. This may result in more than a factor of two variation, but the problem can't be solved without some assumption about independence or not. This is the simplest.

Consider the probability that three disks fail such that one is a data disk, and the other two are the corresponding row and column parity disks. There are 400 possible data disks, and for each such disk, one row-parity disk and one column-parity disk. Let P_3 be the probability that such a three disk failure occurs in a 24 hour period:

$$400 * \left(\tfrac{24}{500000}\right)^3 \left(1 - \tfrac{24}{500000}\right)^{437} \quad \leq \quad P_3 \leq 400 * \left(\tfrac{24}{500000}\right)^3$$

where 437 comes about as $440 - 3$, the number of of disks that don't fail if exactly three disks fail in the configuration described above. Noting that

$$\left(1 - \tfrac{24}{500000}\right)^{437} \quad > \quad 1 - \tfrac{437*24}{500000} \quad \approx \quad 0.98$$

I'll assume

$$P_3 \quad \approx \quad 400 * \left(\tfrac{24}{500000}\right)^3$$

without breaking the factor of two rule.

If any four data disks in a rectangle fail in the same 24 hour period, this is also irrecoverable. Let P_4 denote this probability. By arguments like those above

$$P_4 \quad \approx \quad \tfrac{400*19*19}{4!} \left(\tfrac{24}{500000}\right)^3 \quad \ll \quad P_3$$

So, I'll ignore failures of four or more disks.

The expected number of days between unrecoverable failures is

$$1/P_3 \quad = \quad 22605613426 \text{ days} \quad \approx \quad 62 \text{ million years}$$

Grading: Accept any reasonable assumptions. They don't have to give the mathematical details behind the assumptions. For example, they can say that failures of four or more disks are much less probable than failures of three disks, and never derive anything for the three disk case.

One point off for failing to note that there are four disk failures not covered by the three disk scenario above.

If they got the wrong answer for part (b), then they should derive something consistent with that answer here.

Half credit for observing that the failure rate goes as the cube of the individual disk failure rate, and not the square as in traditional RAID even if they completely messed up the rest of the calculations.

7. (17 points) An argument for simple architectures such as CMP and DIF is that the simpler design should support a higher clock rate. On the other hand, processors such as the Pentium-4 show that a complicated superscalar can achieve high clock rates by making the pipeline deep enough. One consequence of deep pipelining is a higher branch mis-predict penalty. This problem explores the effect of branch mis-predicts.

   (a) (3 points) Using the data from the paper "The Case for a Single Chip Multiprocessor," what is the average instructions-per-cycle achieved by the two-issue superscalar and the six-issue superscalar for the nine benchmark programs that the authors considered?

   (b) (12 points) The paper doesn't report branch mis-predictions. For simplicity, assume that both processors achieved perfect branch prediction for the data reported in the paper. Assume that the 2-issue machine has a relatively short pipeline with a three cycle branch mis-predict penalty. Assume that the 6-issue machine has a long pipeline with a twenty cycle branch mis-predict penalty. Assume that 12% of instructions are branches or indirect jumps (i.e. require prediction). Assume that both processors operate at the same clock rate.

   For what branch mis-predict rate do the two machines have the same performance?

   (c) (2 points) When the mis-predict rate increases, does the ratio

$$\frac{\text{CMP performance}}{\text{super scalar performance}}$$

   increase or decrease?