Architecture

4th September 2002

2002

A1

Suppose you have a computer, say the DLX, with 5 phases: Instruction Fetch(IF), Instruction Decode(ID), Execution(EX), Memory(MEM), and Write-Back(WB). You are considering pipelining the execution of these stages to improve performance. Assume that you will have a 5-stage pipeline with one stage for each of the above phases. Also assume only an integer pipeline (no need to worry about multiple EX stages for floating-point operations.)

1. Give as many reasons as you can for why one cannot expect to obtain a speedup of 5 using this 5-Stage pipeline.
2. Explain what is meant by
   (a) Structural hazards,
   (b) Data hazards, and
   (c) Control hazards

   and the effect of such hazards upon pipeline speedup.
3. For each of the hazards listed in (2), describe the hardware support required by the pipeline in order to minimize the effects of such hazards. If multiple techniques are available, describe any one technique in detail for each of the hazards types. What are the strengths and weaknesses of the techniques described by you?

A2

1. Distinguish between the precise and imprecise interrupts. Why is it desirable to have precise interrupts in a pipelined processor? What is the problem that precise interrupts solve? Why are precise interrupts difficult to achieve?

2. What is meant by "speculative execution"? What are the potential benefits of this approach? Construct a concrete example to explain your answer.

3. What our problems does speculative execution cause for providing precise interrupts?

4. What is the reorder buffer, what does it contain, and how is it used? How does a reorder buffer help to implement precise interrupts in the presence of speculative execution? How is a reorder buffer similar to the reservation stations of Tomasulo's algorithm, and how is it different?

A3

1. 12 pt

   (a) Explain 3 cache optimization strategies.

   (b) Explain a strategy in detail where CPU is involved in the cache miss penalty.

   (c) Suppose that in 500 memory references there are 50 misses in the first-level cash and 10 misses in the second-level cache. What are the local and global missrates for first-level and second-level caches?

2. 13 pt

   (a) Explain 3 techniques to improve memory bandwidth.

   (b) What is the miss penalty for cache block of 4 words with the following performance of the basic memory organization?

      - 4 clock cycles to send the address
      - 32 clock cycles for the access time for word
      - 4 clock cycles to send a word of data

   (c) Give the clock cycle per instruction(CPI) for 16-bit bus and memory, 32-bit bus and memory with no interleaving, and 32-bit bus and memory with interleaving,. In addition to (2) above, assume that

      - Memory accesses per instruction is 1.4
      - Average cycles per instruction (ignoring cache misses) is 2
      - Miss rate is 4% for 16-bit block size and 2% for the 32-bit blocks size.

A4

1. 15 pt

   (a) What is Amdahl's law?

(b) Explain how it can be used to predict speedup for a parallel processor?

(c) Explain other factors that influence the performance of applications on parallel machines?

2. What is multiprocessor cache coherence? Explain 2 different protocols for enforcing this coherence.

2001

A1

1. Explain, with the help of a brief example, how increasing the cache block size can increase the number of coherence misses in centralized shared memory machines.

2. Show, by example, how contention can cause latency of a write-update cache coherence protocol to be higher than that of a write-invalidate protocol.

3. Consider a two-way set associative cache. Show, by a clear example, call increasing the cache block size can increase the number of conflict misses.

4. Explain a strategy in which a large number of the registers can be effectively utilized, without increasing instruction length or the time required for procedure calls.

5. DLX supports displacement addressing mode in which a register number R_i and an 16-bit offset n is used for addressing a memory location M(R_i + n). Suppose we change this instruction set to exclude displacement addressing mode, and to include register indirect M(R_i). Explain how this change can influence the ability to perform instruction scheduling and loop unrolling.

6. Suppose we change a system architecture to allow issuing 8 instructions per cycle, instead of previously supported 4. List what additional resources may be required in the system to allow it to benefit from the increased multiple issue ability, clearly explaining why.

A2

1. you are given a system with the following parameters:

   - The clock speed is 100 MHz
   - the CPI of the system, after including the memory stalls, is 4
   - each instruction generates an average of two memory accesses
   - 30% of memory references are writes

- Each cache block is 8 4-byte words; the entire block is read on a miss
- Cache miss rate is 5%
- Cache uses write-allocate
- Assume 25% of cache blocks are dirty at any given time.

Calculate the total bankwidth with requirements of the memory bus, if the cache is a) write-back, and b) write-through.

2. Suppose we have a 2 processor cache coherent shared memory multiprocessor. The size of main memory is 64 bytes, and each processor has a 16 byte cache. Cache are directly mapped, with a block size of 2 bytes. Given the following sequence of read and write operations (byte numbers) from the two processors, determine if each operation is a

   1) cache bit,

   2) cold cache miss,

   3) conflict cache miss,

   4) capacity cache miss,

   5) true coherence cache misses, or

   6) false coherence cache miss.

   Assume a write back, write-allocate cache.

   Read 0 from P0, Read 8 from P0, Read 1 from P0,

   Read 17 from P0, Read 0 from P0, Read 0 from P1,

   Read 1 from P0, Write 8 from P1, Read 9 from P0,

   Read 9 from P1, Write 8 from P0, Read 8 from P1.

A3

1. Explain how exceptions make implementations of a pipelined architecture harder. Describe different classifications of exceptions, including synchronous vs asynchronous, user requested vs coerced, user maskable vs user non-maksable, within vs between instructions, and resume vs terminate.

2. Explain the following models of consistency, using brief examples to illustrate the differences between them: sequential consistency, total store order (TSO), partial store order, weak ordering, and release consistency. Also describe how each of these models influences the performance and programmability of a shared memory multiprocessor.

A4

1. Discuss the impact of each of the following factors on the branch penalty for conditional branches and for unconditional branches in a pipelined processor:

   (a) The extent and amount of pre-fetching of instructions in the processor

   (b) The processor architecture being VLIW contrasted with superscalar

   (c) The extent to which the processor has "speculative" execution of both instruction streams following a conditional branch.

2. As the chief design architecture of a new pipelined superscalar processor, one of your tasks is to decide the strategy for handling branch instructions. One of your colleagues on the design team is pushing for use sophisticated, expensive branch prediction techniques. Another colleague opposes this approach by asserting that "Correct prediction of a conditional branch outcome merely changes the branch penalty from that of a conditional branch to that of an conditional branch.

   Do you agree or disagree with the assertion of the second colleague? Give logical arguments to support your case.

3. Suppose a pipelined processor has an 8-stage instruction execution pipeline, with each stage taking one cycle. 10% far of all instructions are unconditional branches and 20% are conditional branches. Conditional branches are "successful" (jump to the target) 80% of the time on average. You may assume that there are around no delays due to other causes (e.g., data hazards).

   The penalty for an unconditional branch is 2 cycles. For a conditional branch, if the target is predicted, the penalty for an incorrect prediction is 6 cycles while that for a correct prediction is 2 cycles. If the next sequential instruction is predicted, the penalty for incorrect and correct predictions is 8 and 0 cycles and respectively.

   (a) Suppose the strategy is to always predict that conditional branches will go to the target. What will be the effective CPI (cycles per instruction) of the processor?

   (b) Suppose dynamic prediction is being used. What should be the minimum success rate of the prediction strategy so that the instruction execution rate is higher than for strategy (a)?

2000

A1

1. Explain the limitations of using clocks-rate and MIPS measure to evaluate the performance of a computer system.

2. State the difference between stack based, accumulator based, register based, and load-store architectures.

3. What is the difference between static issue (VLIW) and the dynamic issue (Superscalar) processors? Compare these two designs with the following four considerations.

   (a) Complexity of the hardware

   (b) Complexity of the compiler

   (c) Code Density

   (d) Backward Compatibility

4. Give examples to show RAW, WAW, and WAR dependencies in code sequences. Explain what instruction set design features will cause WAW hazards and WAR hazards.

A2

Consider the implementation of pipelined DLX, a simple instruction set described by Hennessey and Patterson. For integer operations, there are 5 pipeline stages Instruction Fetch(IF), Instruction Decode(ID), Execution(EX), Memory(MEM), and Write-Back(WB). Note that in DLX, add R1, R4, R5 means add R4 and R5 and store in R1. Floating point multiply and addition operations spend 5 cycles in the EX stage. Full forwarding is done to reduce the number of data hazards. Also, assume that there are no structural hazards in the system. A single delayed branch is used to mask stalls due to control hazards.

Consider the following code sequence:

```
for (i=0; i< 1000; i++)
    Y[i] += s * X[i]
```

This loop is implemented in DLX as follows:

```
loop: LD F0, 0(R1)

    MULTD F0, F0, F2
    LD F0, 0(R2)
    ADDD F4, F0, F4
    SUBI R1, R1, 8
    SUBI F2, R2, 8
    BNEZ R1, loop
```

1. Schedule the instructions to minimize the number of data and control hazards, without performing any loop unrolling. Calculate the CPI of the resulting execution of the loop.

2. Now, unroll the loop so that each new iteration does the work of two iterations of the original loop. Schedule the instructions to minimize the number of data and control hazards and calculate the CPI of the resulting execution of the loop. Use the number of instructions in the original loop as the basis for calculating the CPI.

3. Now, suppose the machine has superscalar abilities, with the following features:

   (a) Up to 2 instructions can be started in the same cycle, provided they are aligned at the 65 bit boundary.

   (b) One of the instructions can be a load, store, branch, or integer ALU operation, the other instruction can be a floating point operation.

   Unroll the loop so that each new iteration does the work of four iterations of the original loop (i.e., the loop used in part a)). Schedule the instructions to maximize the use of superscalar abilities and minimize the number of data and control hazards, Calculate the CPI of the resulting execution of the loop. As in part b), use the number of instructions in the original loop as the basis for calculating the CPI.

A3

1. (12 pt) One approach to reducing misses in a cache is to prefetch the next bloc. A simple but effective strategy is: one block "i" is referenced, make sure block "i+1" is in the cache, and if not, prefetch it.

   (a) Do you think this automatic prefetching strategy is more or less effective with the increasing block size? Why?

   (b) Is it more or less effective with increasing cache size? Why?

   (c) Is it more or less effective in an instruction cache compared to a data cache? Why?

   (d) Is it more or less effective in a TLB compared to a regular cache? Why?

2. (13 pt) Compare and contrast each of the following schemes, assuming that each line can hold either the one instruction or one data item.

   (a) A fully associative cache of size "2n" lines using the LRU replacement policy.

   (b) Separate instruction and data caches, both fully associative using the LRU replacement policy, and each of size "n" lines

   (c) An instructions look-behind buffer that holds the "n" most recently executed instructions with an associative search to determine whether a desired instruction is present in the buffer, and a fully associative cache of size "n lines" (for both instructions and data) using LRU replacement policy.

Compare the above schemes with respect to performance, complexity, and types of programs (eg., code structure or data access patterns) that are speeded up by each scheme.

A4

1. (15pt) Describe a write-invalidate write-back cache-coherence protocol for centralized shared memory machines in which four states are used: invalid, shared (read only), exclusive (read/write), and clean private (read only). Draw the state transition diagram, showing the transitions and actions taken. Explain each transition taken in 1-2 sentences, classifying them on the basis of the initial state.
2. (4 pt) Explain why a write-update or write-through coherence protocol are not commonly implemented, especially for larger centralized shared memory configurations.
3. (6 pt) What is Amdahl's Law? What does it tell us about the maximum speed up that can be attained by a parallel processor? Give an expression for the speed up as stated by the law.

1999

A1

1. Explain the term "backward compatibility" as it applies to instruction set by designs. Give 2 examples where can certain kind of architectures have been more successful because of backward compatibility.
2. Briefly explain accumulator, stack, and register based architectures, clearly marking their relative advantages and disadvantages. Give possible reasons that Java Virtual Machine(JVM) is stack based and not register based.
3. State any 3 properties of any instruction set that makes it easier to pipeline to.
4. What is a load-store architecture? From the currently available chips, give one example of a load-store design and one example of a design that is not load-store.

A2

1. Explain the differences between centralized shared memory, distributed shared memory and message passing architectures.

2. Explain the significance of coherence protocols for centralized shared memory and distributed shared memory machines. Also explain why a coherence protocol is not required for message passing machine.

3. Explain the difference between snooping and directories based a coherence protocols. State which one of this is more suitable for centralized shared memory system and which one is more successful for distributed shared memory systems?

4. Describe all write-invalidate, write-back snooping coherence protocol for centralized shared memory system. Assume that 3 states are possible for any cache block: invalid, shared, and exclusive. Draw the coherence diagram and explain each transaction in one sentence.

A3

1. Give an example of each of the following types of interactions between the operands of successive instructions:

   Read/Write, Write/Read, and Write/Write

   For each of these scenes interactions, explain the effect on the performance of a pipelined processor executing these instructions.

2. Described as clearly as you can the technique of internal forwarding. In this technique, how are operand interactions detected and what action is taken to reduce their impact on the performance of the processor? For each of the 3 interactions in part (1), explain how internal fording eliminates or reduces potential delays.

3. How does dynamic scheduling help in further eliminating or reducing delays due to the dependences between instructions.

A4

A computer system that uses Virtual Memory with Paging uses a Translation Lookaside Buffer (TLB) to speed up address translation. It also has a cache which uses physical memory addresses. Both Virtue and Physical addresses (byte-addressable) are 32-bit long, and page size is 4 kilobytes. The TLB can store 16 page table entries, where each entry is 4 bytes long. The TLB uses a direct-mapped organization. The cache is of size 1 kilobytes and is set-associative with four frames, each containing 64 lines of 4 bytes each.

1. Show all the different fields into which a Virtual Addresses and a Physical Address are divided for the purpose of looking up the TLB and the cache respectively. Clearly indicate the position of each field in the address, its size, and the purpose it is used for.

2. List in detail all the steps required to fetch a data item in this system, given the Virtual Address of the item. Use a diagram to show how the different views of the Virtue and Physical Addresses are used in the various steps. Be sure to consider all possible cases: hit/miss in the TLB, hit/miss in the cache, etc.

1998

A1

1. Comment on the issues involved in using virtual addresses in a cache as compared to using physical addresses.

2. State and explain briefly and 2 techniques for achieving higher main memory bandwidth. Compare and contrast between the 2 techniques.

3. 13 pt

   (a) Specify all the portions of a cache address assuming 32-bit physical address, 32-byte cache block site, and a single cache of the size 1024 kilobytes. Consider the following 2 cases: direct mapping, 4-way associative.

   (b) What are the possible advantages/disadvantages of increasing the associate unity in the set-associative cache organization?

A2

1. State and explain amdahl's Law. In the same context, comment if superlinear speed up is possible in real situations.

2. What are the limitations in achieving a linear speedup in the real parallel machine for real application?

3. 12 pt

   (a) Compare and contrast the centralized the shared-memory of architecture vs. distributed shared-memory architecture.

   (b) Explain the use of multiple caches in centralized the shared-memory architecture for multiprocessors. What problems does the use of the cache cause? How are they resolved?

A3

1. In the context of pipelined processors, explain the terms dynamic scheduling and static scheduling, clearly describing the role that the compiler and the hardware plays in each.

2. What is the difference between static issue(VLIW) and dynamic issue (super scalar) processors? Compare these 2 designs with the following for considerations:

    (a) Complexity of the hardware

    (b) Complexity of the compiler

    (c) Code density

    (d) Backward compatibility

3. Explain how loop unrolling can help in improving performance in pipelined and multiple issue machines.

A4

Consider the simplest the implementation of pipelined integer DLX, an instruction set described by Hennessy and Patterson. There are 5 pipelines stages, Instruction Fetch(IF), Instruction Decode (ID), Execution (EX), Access Data Memory(MEM), and Write-Back into the Registers (WB). Assume that all the registers are written only during the last cycle in the pipeline. Also, note that in DLX, add R1, R3, R5 means add R4 and R5 and store in R1. Load R3, O(R1) means that O is added to R1, and the result is used as the source address; R3 is the destination of the load. No structural hazards are possible.

Consider the following code sequence.

```
Load R3, 0(R1)
Add R5, R3, R4
Load R6, 0(R2)
Add R7, R6, R4
```

1. Initially, assume that no forwarding is possible. Draw the resulting pipeline timing chart and calculate the number of cycles it takes to finish in the above code sequence.

2. Assume that the forwarding is done in the implementation of the machine. Show the resulting pipeline timing chart and calculate the number of cycles required.

3. How can the performance of the above code be improved through compiler re-ordering of the instructions? Show the new pipeline timing chart and calculate the number of cycles required.

1997

A1

1. Consider the following code sequence

```
for (j=0; j<4; j++)
  for (i=0; i<256; i++)
    x[i][j] += y[i][j]
```

Assume that

- The arrays x and y are dimension 256*4. The storage allocation is row major.
- Each data item is 4 bytes.
- Cache block size is 32 bytes.
- Total cache size is 8 kilobytes.
- The cache is fully associative with least recently used replacement policy.
- The cache is write-back, with write-allocate.

(a) Determine the number of cache misses during the execution of the above loop-nest. Assume that no elements of x or y are in cache initially.

(b) Based upon your result in the previous part, will there be any advantage to applying the loop transformation to the previous code?

(c) How will you answer be part (b) changed if the cache was directly mapped?

2. Explain the Amdashl's law? How does Amdashl's law predict performance for the parallel machines? What on the other factors which influence the performance of applications on parallel machines? Explain the effect of these factors separately for message passing machines, centralized shared memory machines and distributed shared memory machines (8+8+3+3+3).

3. 25 pt

   (a) Explain the difference between RISC and CISC architecture. State 2 important reasons for the emergence of RISC designs.

   (b) State the difference between stack based, accumulator based, register days and load-store architectures. Give 1 example of a processor which is registered based, but not load-store.

   (c) State 2 important considerations in design of an instruction set, from the view-point of pipe Line implementation and performance.

4. Consider the simplest implementation of pipelined integer DLX, a simple instruction set described by Henson and Patterson. For integer operations, there are 5 pipeline stages Instruction Fetch(IF), Instruction Decode(ID), Execution(EX), Memory(MEM), and Write-Back(WB). Assume that all

the registers are written only during the last cycle in the pipeline. Note that in DLX, add R1, R4, R5 means add R4 and R5 and store in R1. For any of the ALU operations (like add or sub), the operands are read during the decode(ID) stage and the calculations are performed during the EX stage.

Consider the following code sequence:

```
add R1, R4, R5
sub R6, R1, R5
add R7, R1, R6
add R8, R4, R1
```

(a) If we stall the pipeline whenever there's a date hazard, how many cycles with it take to complete the above instructions sequence? Draw the resulting pipeline.

(b) Can the performance of the above code be improved by compiler-recordering of the instructions? Redraw the resulting pipeline and calculate the number of cycles required.

(c) Assume that we now have a machine which is able to input data from any pipeline register rather than having to wait till the data is written to the real registers. Show the new pipeline and calculate the number of cycles required.