# 國立交通大學試題紙

九十八學年度第一次
博士班資格考

科目：計算機結構(A)　　日期：99 年 1 月 27 日　第 1 頁 共 2 頁

請"✓"明　✓不可看書　可看書。This is a CLOSED-/~~OPEN-~~ book examination.

* 請將答案依題號順序寫入答案卷。Present your answers in strict problem order.
答題時字跡需工整，否則不予計分。Write your answers legibly; otherwise you will get zero score.

1. (8 %) We discuss the performance of a uniprocessor. The authors of our textbook have pointed out the significance of the quantitative approach, and suggested an equation to calculate the CPU time in Section 1.9.
   (a) (2%) List that equation.
   (b) (6%) The equation used a few parameters, and changing the parameters can improve performance. For each parameter, list methods that can improve it, respectively. (List no more than threefor each parameter, but please pick the important ones first.) Your answer should include what that method means, when the method is applied by what mechanism, and if the method could be harmful to other parameters in the equation.

2. (6 %) For every component in a computer system, its performance can be viewed in terms of latency as well as throughput. In answering the following questions, let's take a microprocessor as the system component, for example:
   (a) Does improving latency usually hurt throughput? Why? And use an improvement technique to support your answer.
   (b) Repeat the above, but reverse the roles of latency and throughput.

3. (15 %) Data dependency exists in three forms: true, anti-, and output dependencies. Assume a single processor and a sequential program:
   (a) Which ones can cause stalls in sequential, in-order, non-pipelined execution?
   (b) Which ones can cause stalls in sequential, in-order, pipelined execution?
   (c) Which ones can cause stalls in sequential, out-of-order, pipelined execution?
   (d) Which ones can cause stalls in superscalar, in-order, pipelined execution?
   (e) Which ones can cause stalls in superscalar, out-of-order, pipelined execution?
   (In your answers, whenever you give an answer, give an example to show why that is true.)

4. (15%) Answer the following questions about Tomasulo's algorithm used in IBM 360/91. Similar design is used in MIPS floating-point unit. Make your answers as concise and precise as possible.
   (a) How is register renaming implemented here?
   (b) How is data forwarding implemented here?
   (c) How does this unit handle exceptions? Specifically, are the exceptions precise?
   (d) How many results can be calculated per cycle here? (You need not know the exact number, but just tell me whatever units can produce results simultaneously.)
   (e) How many instructions (or operations) can complete (call it commit if you like) simultaneously?

5. (6%) Branches hurdle instruction fetch and limit instruction-level parallelism. Many branch handling schemes are introduced in the text.
   (a) Show one example of how static branch handling works in sufficient detail, so that I can see that it is really implementable.
   (b) How does correlating predictor work?
   (c) Repeat (b) for tournament predictor.

# 國立交通大學試題紙

九十八學年度第一次
博士班資格考

科目：計算機結構(B)　　日期： 99 年 1 月 27 日　第 1 頁 共 3 頁

請 "✓" 明　✓不可看書　可看書

* 請將答案依題號順序寫入答案卷

答題時字跡需工整，否則不予計分。Write your answers legibly; otherwise you will get zero score.

1. (8 %) The following questions describe two variants of a processor that are otherwise identical. In each case, circle "**Yes**" if the difference should be visible to software in the ISA, and circle "**No**" otherwise. You must also **briefly explain** your reasoning and any assumptions you are making. *Ignore differences in performance.*
   (a) Pipelined processor A has more stages than pipelined Processor B, but both have full bypassing.
   (b) Processor A uses microcoded control while Processor B uses hardwired control.
   (c) Processor A is considered to be a CISC machine while Processor B is a RISC machine.
   (d) Stack machine A has more physical registers to implement its stack than stack machine B.

2. (15%) Mark whether the following modifications will cause each of the categories to **increase, decrease**, or whether the modification will have **no effect**. You can assume the baseline cache is set associative. **Explain your reasoning** to receive credit.

| | Compulsory Misses | Conflict Misses | Capacity Misses |
|---|---|---|---|
| Double the associativity (capacity and line size constant) | | | |
| Halving the line size (associativity and number of sets constant) | | | |
| Doubling the number of sets (capacity and line size constant) | | | |
| Adding a victim cache | | | |
| Adding prefetching | | | |

3. (9 %) Consider two different machines. The first has a single cycle datapath (i.e., a single stage, nonpipelined machine) with a cycle time of 4ns. The second is a pipelined machine with four pipeline stages and a cycle time of 1ns.
   (a) What is the speedup of the pipelined machine versus the single cycle machine assuming there are no stalls?
   (b) What is the speedup of the pipelined machine versus the single cycle machine if the pipeline stalls 1 cycle for 30% of the instructions?
   (c) Now consider a 3 stage pipeline machine with a cycle time of 1.1ns. Again assuming no stalls, is this implementation faster or slower than the original 4 stage pipeline? Explain your answer.

4. (6 %) Assume the cache has a **90%** hit rate, and that the way-predictor is right **75%** of the time there is a cache hit. A cache hit with a correct way-prediction will take **2 cycles**, a cache hit with an incorrect prediction will take **4 cycles**, and a cache miss time (way-prediction irrelevant) will take **60 cycles (hit time + miss penalty)**. Compute the average memory access time of the cache with way-prediction.
   Without way-prediction (the original 2-way set associative cache) has the same hit rate and miss time, but a **3 cycle** hit time. By how many cycles does way-prediction improve the average memory access time?

5. (12 %) The simple, bus-based multiprocessor illustrated in following figure represents a commonly implemented symmetric shared-memory architecture. Each processor has a single, private cache with coherence maintained using snooping coherence protocol. Each cache is **direct-mapped** with four blocks each holding two words. To simplify the illustration, the cache-address tag contains the full address and each word shows only two hex characters, with the least significant word on the right. The coherence states are denoted M. S. and I for Modified, Shared, and Invalid.
   We assume the initial state and memory state as illustrated in following figure. Each part of this exercise specifies a sequence of one or more CPU operations of the form:
   P#: <op> <address> [ <-- <value> ]
   where P# designates the CPU (e.g., P0), <op> is the CPU operation (e.g., read or write), <address> denotes the memory address, and <value> indicates the new word to be assigned on a write operation.
   Treat each action below as **independently** applied to the initial state as given in following figure. What is the resulting state (i.e., coherence state, tag, and data) of the cache and memory after given action? Show only the blocks and memory that change.
   For example, we assume that the first CPU operation: P0: read 100 has completed, the final state of the cache become: P0.B0: (S, 100, 00 00) and no memory change. Where P0.B0: (S, 100, 00 00) indicates that CPU P0's block B0 has the final state of S, tag of 100, and data words 00 00.
   (a) P0: write 130 <-- 98
   (b) P15: write 130 <-- 98
   (c) P0: read 120
   (d) P1: read 110

# 國立交通大學試題紙

九十八學年度第一次
博士班資格考

科目：計算機結構(B)　　日期： 99 年 1 月 27 日　第 3 頁 共 3 頁