

Name _____ ID# _____

Architecture Comprehensive Exam September 26, 2005

The exam is made up of four questions, for a total of 100 points.

Question	Points
1	/ 25
2	/ 25
3	/ 25
4	/ 25
Total	/100

1. **(25 points)** Let's define S_{ooo} as the speedup observed for a baseline out-of-order (dynamically scheduled) processor over a very similar in-order processor. By similar, I mean similar pipeline length, same cache hierarchy, same clock, etc. If we make one of the following changes to both processors, would the ratio S_{ooo} **INCREASE** or **DECREASE** for the two new processors, relative to the original S_{ooo} ? You must explain in detail each of your answers.

(a) smaller L1 data cache (higher L1 data cache miss rate).

(b) smaller L1 instruction cache (higher L1 instruction cache miss rate).

(c) larger L1 data cache miss penalty.

(d) better branch predictor.

(e) longer pipeline (greater misprediction penalty)

(f) wider superscalar fetch and issue bandwidth

(g) support hardware multithreading

2. (25 points) Branch Prediction

For the questions below, assume you have a 5 stage pipeline (FETCH, DECODE, EXE, MEM, WB). In answering the below questions, state exactly in which of the five pipeline stages each of the below structures and their components are (1) accessed, (2) updated, and (3) which stage a branch misprediction is determined. In addition, for this pipeline with the branch predictor given below, what is the branch misprediction penalty?

(A) Add to the 5 stage pipeline a 512 entry 4-way associative entry BTB and show exactly how many bits are used for the index.

(B) Add to the BTB, the following conditional branch predictor. In this design, the conditional branch predictor is only used on a BTB hit. If this is the case, then what is the best choice for predicting conditional branches on a BTB miss given the pipeline above?

The conditional branch predictor has a 4K entry 2-bit table indexed by a global history XORed with the PC. You need to draw (1) all of the logic for this structure and how it interacts with the BTB, and (2) exactly what bits are used and the number of bits used to index each structure.

(C) Show exactly what fields are stored in the BTB above assuming the branch predictor above, and show how the next PC is calculated given all of the information from the BTB in (A) and the branch predictor in (B).

3. (25 points) Cache Tagging

A particular program is to be executed on a processor with L1 data cache, and it has been found that the program only accesses data within the following two disjoint address ranges: [1000,000000 : 1001,111111] and [1110,000000 : 1111,111111] (the processor uses 10 bit addresses). For the particular L1 data cache organization, the 4 most significant bits from the address correspond to the cache tag (in the range denotation above, the tag part of the address has been separated by a comma for your convenience). In order to reduce the cache power consumption, instead of using all of these 4 most significant address bits as tags (as a traditional cache organization would do), the hardware engineers have decided to skip some of the most significant tag bits when accessing the cache.

(Part A) As a first option of this tag-optimized cache architecture, it has been decided that the cache tag arrays would store only the “reduced” tags. The following table shows 6 different configurations where 1, 2, or 3 most significant tag bits have been discarded; both write-through and write-back cache policies are being considered. For each such configuration please specify if the cache would function correctly for this particular program with the address ranges given above. Please explain your answers. For any configurations identified to be functionally correct, please comment on the miss rate compared to the traditional cache organization that would use all of the 4 most significant bits as tags.

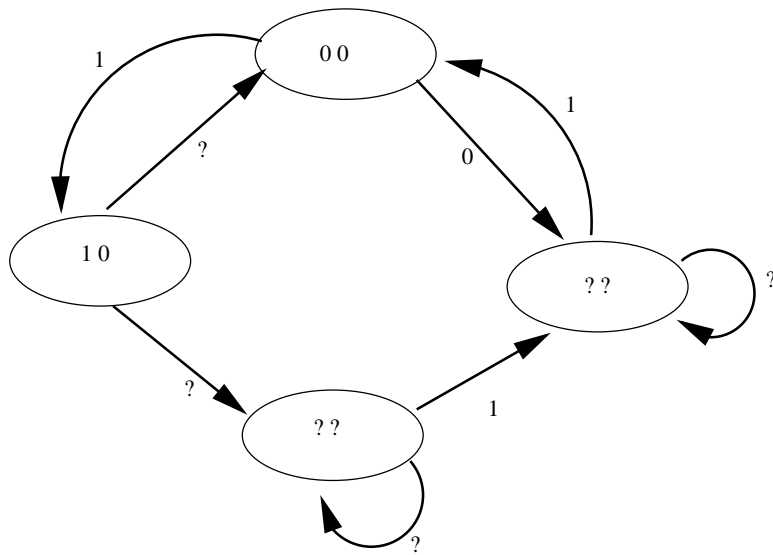
Most significant tag bits skipped	write-through cache	write-back cache
3		
2		
1		

(Part B) As a second option it has been decided that the tag arrays would actually capture the *entire* 4 bits of original tags, but would not use some of the most significant bits for tag comparison when accessing the cache. Consider the same 6 configurations as in Part A. For each such configuration please specify if the cache would function correctly for this particular program with the address ranges given above. Please explain your answers. For any configuration identified to be functionally correct, please comment on the miss rate compared to the traditional cache organization that would use all of the 4 most significant bits as tags.

Most significant tag bits skipped	write-through cache	write-back cache
3		
2		
1		

4. (25 points) Finite State Machines

Consider the following scenario. An FSM was implemented using T-flip flops using the corresponding next state logic. The actual implementation charts were lost; the only thing that partially survived was the FSM state diagram, together with the next state logic. Only one input exists in this FSM and 2 bits are used to represent all 4 states. Following is the partial FSM state diagram that was obtained by using T flip flops.



As the designers who found the next state logic together with the partial FSM did not know what flip flops the machine originally was designed for, they went ahead and used the recovered next state logic with the D flip flop, as D is the most commonly used flip flop. A smart engineer was able to figure out what the machine corresponding to the original got transformed to when the original next state logic got coupled to the D-flip flops. Can you help him finish his job by filling in the missing parts (denoted by "?") in both diagrams.

Hint: Both machines are using the same "next state logic". All the T-flip flops from the previous machine have simply been replaced by D flip flops. What effect will this have?

