**NATIONAL TSING HUA UNIVERSITY**

**DEPARTMENT OF COMPUTER SCIENCE**

## CS 4100: Computer Architecture

# Spring 2018, Final Examination

1. (16%) Consider the pipelined design of the MIPS processor and the code segment shown below.

 **sub $2, $1,$3**

**lw $4,50($2)**

**add $4, $6,$2**

**beq $4,$2,-5**

**sw $4,100($2)**



* + 1. (8%) Suppose at any cycle the register file can perform either read (read from two registers) or write (write into a register), but not both. (a) What type of hazards will this design produce? Use the above code segment to explain when this type of hazards will occur. (b) To resolve this type of hazards, we let write-to-register take a higher priority than read-from-register. Explain how you will handle the IF/ID and ID/EX pipeline registers, respectively, when the hazard occurs, e.g., setting the control signals.
		2. (5%) For the following questions, assume that the register file can support two reads and one write at the same cycle. What is the size of the ID/EX pipeline register, excluding the control signals?
		3. (5%) When **sub** is at the WB stage and **add** at the EX stage, what are the values of the two outputs of the “Forwarding unit”? Give your reasons. (Note: the possible values at each of the two outputs are 0, 1, or 2.)
		4. (11%) (a) When **lw** is at the MEM stage, **add** at the EX stage, and **beq** at the ID stage, what is the output value of “Sign-extend” in hexadecimal? (b) Explain why the data hazard between **lw** and **beq** or between **add** and **beq** CANNOT be resolved by forwarding? (c) How can this data hazard be resolved? Explain your answers.
		5. (5%) It is possible to forward the ALU result from the MEM stage to the ID stage, if **beq** is at the ID stage and it is data-dependent on the instruction at the MEM stage. Draw a diagram to show how the outputs of the register file should be modified to take the data forwarded from the MEM stage. (You need not show the forwarding unit.)
		6. (5%) We want to extend the IF stage to include a branch target buffer (BTB) with branch predictors. Draw a diagram to show how the input to and output from the PC register should be modified to work with the BTB. Explain how your design work so that the next cycle can fetch instructions from the predicted path after a **beq**.
		7. (5%) Explain what happen when **add** causes an overflow exception at the EX stage.
1. Suppose we decide to design a two-way set-associative data cache for the above pipeline. The pipeline uses 32-bit byte addresses and 32-bit words. The data cache consists of 512 sets. Each cache block contains 2 words.
	* 1. (5%) What is the total size of the cache in bits, including the valid bits, tag bits, and data bits?
		2. (6%) Explain how this cache organization exploits spatial locality of reference, and how it reduces compulsory misses.
		3. (8%) (a) For this cache organization, the time required for a read hit and a write hit are different. Explain why. (b) On the other hand, a direct-mapped cache with one work per block has almost the same read hit and write hit time. Give you explanations. (Hint: Do we need to check the old data in the cache on a write hit for this cache?)
		4. (5%) Ideally, the time to perform a cache access should be affected by that access only and not by other accesses. Which write policy, write-through or write-back, is close to the ideal? Give your explanations.
		5. (12%) Suppose we want to perform array addition of an array A[2048] (2048 words) with each of the four arrays: B0[2048], B1[2048], B2[2048], and B3[2048]. Results are accumulated in A[2048]. Assume the program does not access other variables, and arrays A, B0, B1, B2, and B3 are allocated sequentially in the memory by the compiler. (a) How many misses will occur if we perform A+B0, A+B1, A+B2, and A+B3 in sequence? (b) Among the misses, how many are conflict or capacity misses? (c) How will you rewrite the program to eliminate conflict and capacity misses?
		6. (6%) Suppose the **lw** instruction in Question 1 produces an address of 0x004B50E2 at the EX stage. When it enters the MEM stage, the address is used to fetch a word from the memory. Now, assume that the system runs a virtual memory with a page size of 4096 bytes. (a) What is the virtual page number of this address? (b) Explain how the address is translated into the physical address with the help of TLB and page table.
		7. (5%) From Question 2(6) above, will the portion of the address that we use to index a set in the data cache be changed before and after virtual address translation? Can virtual address translation in TLB and data access in cache be performed at the same time, instead of sequentially?
2. (8%) (a) Compare isolated I/O and memory-mapped I/O. (b) How does the CPU know where to execute the corresponding interrupt service routine on an interrupt?