CS3350B, Computer Organization
Assignment 4

March 18, 2024

Submission Instructions

IMPORTANT Before submitting your assignment, it is essential to check the following link: https://bit.ly/3IJLx72. This online document contains all clarifications and updates related to this assignment. The assignment PDF may undergo minor updates in certain circumstances. We ensure the most recent version of the assignment is always available in this link for download. To guarantee your submission is in line with the latest version of the assignment, always check this link before submitting.

Collaboration and Integrity Policy For any question involving calculations, you must provide your workings. While collaboration with other students in the class on general strategies for solving the problems is permitted, each assignment and the answers within must be solely individual work and completed independently. Any form of plagiarism or the use of AI assistance, such as ChatGPT, will be taken seriously and may result in a mark of 0 on this assignment, removal from the course, or more severe consequences.

Single PDF Submission All answers for this assignment must be submitted on OWL as a single PDF file, named "asn4_answer.pdf". Failure to comply with this naming convention will result in a deduction of marks.

LaTeX Bonus A bonus of up to 15% of the total marks for this assignment (not exceeding the maximum mark) will be awarded for using the provided LaTeX template to complete the assignment. The template ensures a high level of readability and formatting consistency. The LaTeX file is provided as a .zip archive and is recommended to be uploaded to Overleaf for compilation into a PDF (you may also use other Latex compiler you like). You're encouraged to utilize the provided tables in the template for a more straightforward completion of your answers. Also, you may include additional figures as necessary. Remember, only the final PDF file needs to be submitted.

Other formart While it is highly recommended to type your answers using the provided LaTeX template or another typing software, clear scans (not photographs) of handwritten work

will be accepted if they are exceptionally legible. Please note that if your submission is difficult to read or interpret, it may be marked incorrect, with no option for re-evaluation.

Exercise

Exercise 1. [10 Marks] Consider the multi-cycle MIPS datapath presented in Figure 1, it shows 4 inter-stage registers: IF/ID, ID/EX, EX/MEM, and MEM/WB. Consider also the control signals presented in the diagram in blue. Assume the ALUOp control signal is 3 bits. Ignore control signals not shown (i.e. the ones controlling forwarding). Determine the minimum size, in bits, of each of the four inter-stage registers. For part marks, ensure to include workings and not just the total values.

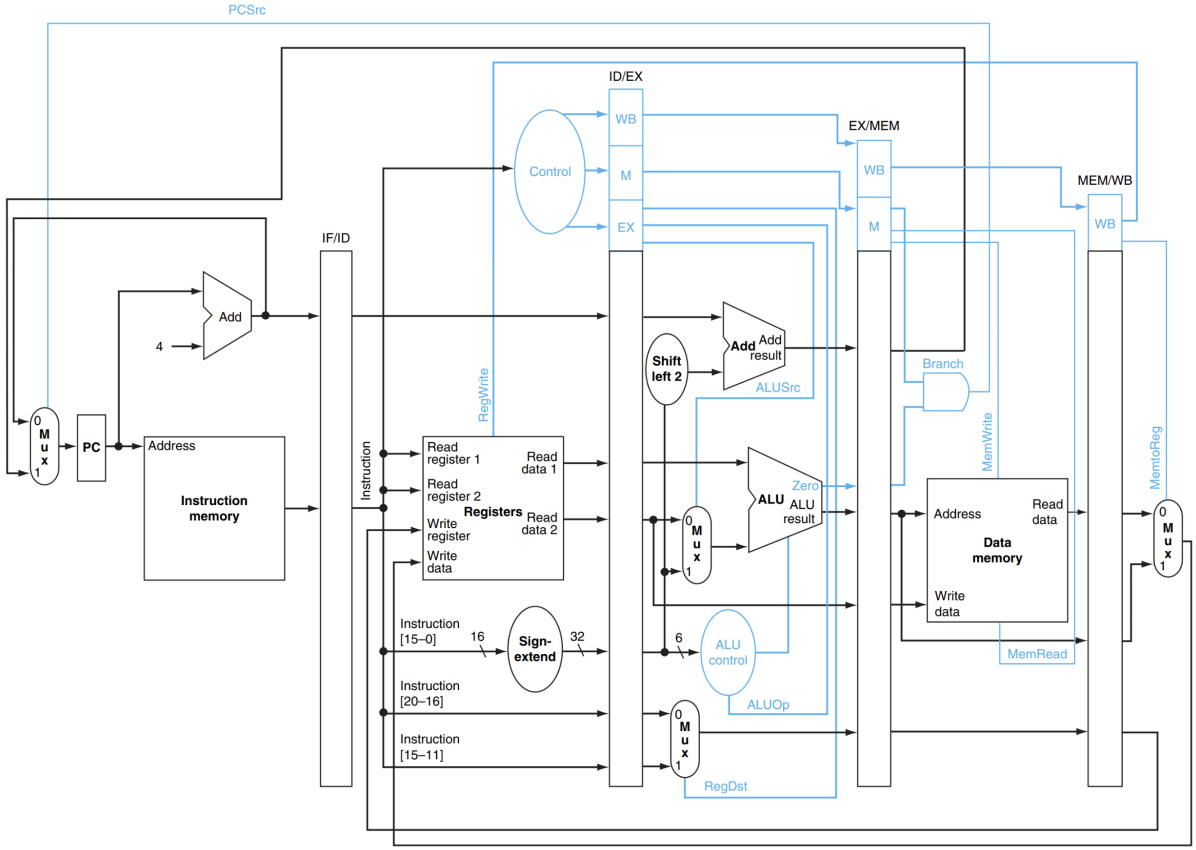

Figure 1: MIPS Datapath with Interstage Registers.

Exercise 2. [10 Marks] Consider a multi-core processor with 4 cores (named P1, P2, P3, P4), each with a dedicated cache. All 4 cores are attempting to read and write to the same cache block. The accesses to that cache block are serialized in the order that they are listed in the following table, from top to bottom.

Use the MESI protocol to complete the following table, showing the state of the cache block in each cache as requests are being processed. In each row, specify the state of the cache block in each core's cache after the request has been satisfied. On a cache miss, also specify all possible data suppliers. Assume that all caches are initially empty. Fill the following table (the first 3 lines are filled as a hint).

op	**P1**	**P2**	**P3**	**P4**	**Supplier**
P1Read	E	-	-	-	Main Mem
P1Write	M	-	-	-	None
P2Read	S	S	-	-	P1
P3Write					
P1Read					
P4Read					
P3Read					
P2Read					
P3Write					
P4Write					
P2Read					
P4Write					

Exercise 3. [10 Marks] Consider the following C code segment where a and b are 32-bit integer arrays of size n.

```
for (int i = 0; i < n; ++i) {
    int t = a[i];
    a[i] = b[i];
    b[i] = t + 2;
}
```

We have a corresponding MIPS instruction sequence, where $s0, $s1, $s2, and $t4 initially store, respectively, the base address of a, the base address of b, n, and $i = 0$.

```
loop:
    lw   $t1, 0($s0)
    lw   $t2, 0($s1)
    sw   $t2, 0($s0)
    addi $t1, $t1, 2
    sw   $t1, 0($s1)
    addi $s0, $s0, 4
    addi $s1, $s1, 4
    addi $t4, $t4, 1
    slt  $t5, $t4, $s2
    bne  $t5, $0, loop
```

Assume that the above MIPS instructions will be executed on a 5-stage pipelined processor. Ignore control hazards and structural hazard. Assume the branch condition is computed in the EX stage.

(a) Draw the pipeline execution diagram (see *L13-HazardExamples.pdf*) for one iteration of the loop: from the first `lw`, up to and including the `bne`. Assume there is no data forwarding. Do not re-order the instructions.

(b) Now, assume all possible data forwarding is used. Draw the pipeline execution diagram for one iteration of the loop: from the first `lw`, up to and including the `bne`. For each instance of data forwarding annotate your pipeline diagram (say, with arrows, colors, and/or footnotes) to indicate the source and destination of the forwarding; also say what kind of forwarding it is. Do not re-order the instructions.

(c) Apply loop unrolling (as well as instruction re-ordering, if you like) on the MIPS code segment so that the unrolled code is equivalent to two iterations of the original. Write out your final MIPS instruction code. Make sure to use offsets appropriately to avoid unnecessary instructions. You may assume n is a multiple of 2.

(d) Consider a 2-issue extension of MIPS. That is, a VLIW extension of MIPS where two instructions occur in each issue packet. The issue packet has the following format: the first instruction must be arithmetic or branch, and the second instruction must be a data transfer instruction (`lw` or `sw`). Still assume all types of forwarding.

Using the code of your unrolled loop of part (c), statically schedule one iteration of the (now unrolled) loop to run optimally on this 2-issue machine. You may wish to modify your code from part (c) to have a different order and use additional registers to accomplish this task. Consider using the following table as a starting point and to help guide you through the process

	ALU or branch	Data transfer	CC
loop:			1
			2
			3
			4
			5
			6
			7
			8
			9
			...

Exercise 4. [10 Marks] Consider a multi-core processor with 2 cores, named P1 and P2. Each core has a dedicated cache with the following characteristics:

- 2-way set associative and a 16-byte capacity;
- is initially empty;
- follows the MESI snooping protocol;
- follows write-back and write-allocate protocols; and
- follows a pseudo-LRU replacement policy where
  - (i) empty cache lines in a set are filled first, then,
  - (ii) if there are any invalid cache lines in a set replace them, then,
  - (iii) if no invalid cache lines are present, follows a typical LRU replacement policy.

Given the following list of serialized memory byte address accesses by the cores, determine:

(a) whether each access results in a cache hit, cold miss, conflict miss, capacity miss, true share miss, or false share miss;

(b) the data stored in each cache after all addresses in the list have been accessed; and

(c) the MESI state of each cache block after all addresses in the list have been accessed.

Time	Memory Access	Hit/Miss Type	Time	Memory Access	Hit/Miss Type
1	P1 Reads 5		11	P2 Writes 9	
2	P2 Writes 8		12	P2 Writes 10	
3	P1 Reads 9		13	P2 Reads 2	
4	P1 Writes 14		14	P1 Writes 7	
5	P1 Reads 3		15	P1 Reads 8	
6	P1 Writes 12		16	P1 Reads 4	
7	P2 Reads 6		17	P2 Reads 12	
8	P2 Reads 17		18	P2 Reads 7	
9	P1 Reads 20		19	P1 Writes 2	
10	P2 Reads 4		20	P1 Reads 11	

To answer part (a) use the above table. To answer parts (b) and (c), use the below tables.

P1 Cache				P2 Cache			
Set	Cache Block Data		State	Set	Cache Block Data		State
0				0			
1				1			
2				2			
3				3			