Homework 2

CDA 4102/5155: Fall 2025

Due: September 23, 2025 **11:30 pm** **Total Points:** 20 points

You are not allowed to take or give help in completing this assignment. Submit the PDF version of the submission in e-learning before the deadline. Late submission (by email attachment to nkim2@ufl.edu) is allowed (up to 24 hours) with a 20% penalty (irrespective of whether it is late for 10 minutes or 10 hours). No grades for late submissions after 24 hours from the deadline. Handwritten (scanned PDF) submissions will NOT be accepted. Please write it in LaTeX or Microsoft Word and convert it to PDF. Please do not take any help from LLM (e.g., ChatGPT) or any other sources. *Please do not include the questions in your solution (PDF) since it affects the plagiarism checker.* Please include the following sentence on top of your submission (PDF):

I have neither given nor received any unauthorized aid on this assignment.

---

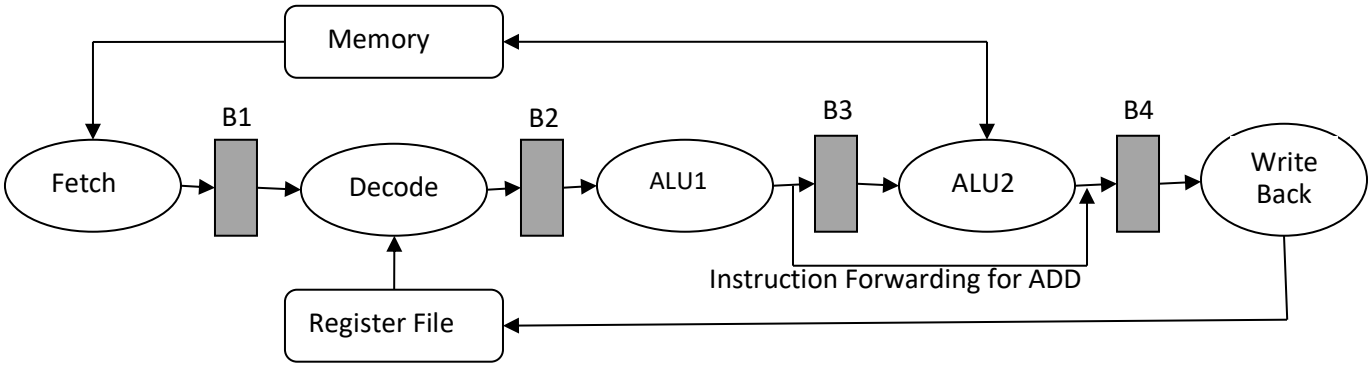
1. **[5 pts]** Write a sequence of RISC-V assembly instructions to count the number of '1's in the value stored in x1. Store the result in x2. For example, if we provide x1 as 0xF004ABCD, then x2 must be 0xF after executing your code. Please use only four registers (x0, x1, x2, and x3). Assume that x0 is always 0. Please do not initialize x1, assume that some negative value (less than 0) is already there (leftmost bit is 1). Please use only one branch (conditional or unconditional) instruction. [*Please use the instructions from riscv-ISA.pdf in the course website (eLearning)*].

2. **[5 pts]** You need to show the instruction-by-instruction computation of the following sequence of RISC-V instructions assuming **4-bit 2's complement arithmetic**. Assume that all 32 registers (x0 – x31) are initialized to 0. Moreover, each K-th memory location has been initialized with K (if K is even) or –K (if K is odd). For example, if you read the contents of memory addresses 12 and 13, it will return the value of 12 and –13, respectively. Perform the computation of each instruction using 4-bit 2's complement arithmetic. Please show the content of the registers or memory values modified by each instruction as **signed decimal values**. You do not have to show the conversion procedure to 4-bit 2's complement arithmetic or actual binary add/shift. However, you need to **show the contents of registers and memory locations** that are affected by an instruction. For example, in case of the first instruction, show only the content of x1 after the computation. Please show major steps (e.g., what is the 2's complement value, what is the address for the load instruction, etc.). Here, lw, sw, add, and sra are load, store, add, and shift-right-arithmetic operations, respectively. Please print as "x2 = 3" or "x2 = -3" for registers, and "Memory[13] = 13" or "Memory[13] = -13" for memories.

```
lw x1, 3(x2)
lw x3, 4(x1)
add x4, x1, x3
srai x5, x4, 2
sw x5, 3 (x0)
```

3. **[5 pts]** Consider a simple 5-stage pipeline, where each stage takes different times to complete: IF 3.2 ns, ID 2.3 ns, EX 4.5 ns, MEM 3 ns, and WB 1 ns. There is 0.5 ns delay associated with other design constraints (latches and clock skew) due to pipelining. Assume that we would like to execute **only 100 instructions**. Compute the ***clock period***, ***speedup***, ***efficiency***, and ***throughput*** for this simple pipelined processor. Please show major steps. [*Slide 10 of pipelining.ppt may be helpful for this problem.*]

4. **[5 pts]** Consider the following processor with 5 pipeline stages: Fetch, Decode, ALU1, ALU2 and WriteBack. Each unit consumes only one clock cycle except ALU2 which consumes 2 cycles. Please note that once an instruction enters ALU2 unit, it remains busy for 2 cycles, i.e., the next instruction cannot enter ALU2 in the next cycle. The shaded rectangular boxes between units are buffers to hold instructions. The first three buffers (B1, B2, and B3) can hold only one instruction. B4 can hold two instructions. Buffers do not consume any clock cycles. Fetch unit fetches one instruction per cycle, and every unit executes (or decodes, writes back, etc.) the instruction from the input buffer (if any) and writes to the corresponding output buffer at the end of the cycle. There are no data forwarding paths. All source registers are read at Decode unit and all destination registers are written by WriteBack unit. There is one instruction forwarding path from ALU1 to WriteBack. This forwarding path is used by the single-cycle ALU operations (i.e., ADD does not use the ALU2 stage). Only three cycle operations use both ALU1 and ALU2. There are three three-cycle operations: multiplication (MUL), load (LD) and store (ST). The MUL operation needs both ALU1 and ALU2 stages (total three cycles) to perform multiplication. The LD/ST operations use ALU1 stage (one cycle) to perform address calculation and ALU2 stage (two cycles) to perform actual load/store from/to memory. Decode stage handles all hazards (data and structural) and stalls itself and the Fetch unit, if required. WriteBack to the register file and the corresponding data read from the register file cannot happen in the same cycle. Please pay attention to the fact that each buffer has only one entry (except B4), therefore, it may be necessary to stall a specific unit and its predecessors at a specific cycle to avoid data loss or to ensure functional correctness. Note that Memory unit has only one port. Therefore, both read and write cannot happen at the same time. Also, note that the same Memory unit is used by Fetch and ALU2. Therefore, fetching an instruction will lead to structural hazard if load or store is using it at the same time. Finally, Writeback unit can write up to two instructions (out-of-order) per cycle.



Complete the following entries for each instruction for the architecture shown above. This architecture uses a **scoreboard**. Each entry in the table should indicate the clock cycle number when that task was **done/completed** for that instruction. Also, please indicate hazard information (if applicable), such as RAW, WAR, WAW, Control, and Structural, in the boxes when you are delaying any activity.

ADD R1, R2, R3 ➔ R1 = R2+R3; MUL R1, R2, R3 ➔ R1 = R2 x R3; LD R1, 8(R2) ➔ R1 = M [R2+8]; ST R1, 8 (R2) ➔ M [R2+8] = R1

Instructions	**Fetch**	**Decode**	**ALU1**	**ALU2**	**WriteBack**
MUL R1, R2, R3	1	2	3	4-5	6
LD R4, 8(R1)	2	7 (RAW)			
ST R5, 8(R7)					
ADD R8, R5, R6					
MUL R9, R8, R5					