# CS450/650 Computer Architecture – HW2
Winter 2026

Total credit: 130 points
Submission site: Crowdmark
Due: 11:59 PM EST, February 5, 2026

## 1. Performance Evaluation (10 points)

Your job is to evaluate the potential performance of two processors, each implementing a different ISA. The evaluation is based on its performance on a particular benchmark. Millions of Instructions per Second (MIPS) and average execution time per instruction (second) are used as the metric for benchmarking. On the processor implementing ISA A, the best-compiled code for this benchmark performs at the rate of 3 IPC. That processor has a fixed clock frequency of 500 MHz. On the processor implementing ISA B, the best-compiled code for this benchmark performs at the rate of 7 IPC. That processor has a fixed clock frequency of 200 MHz.

1. What are the performance in MIPS and the average execution time per instruction (second) of the processor implementing ISA A? (2 points)
2. What are the performance in MIPS and the average execution time per instruction (second) of the processor implementing ISA B? (2 points)
3. Which is the higher performance processor: A, B, or don't know. Explain your answer. (6 points)

## 2. ISA Tradeoffs (60 points)

Your task is to compare the memory efficiency of five different styles of instruction sets for the code sequence below. The architectural styles are:

1. A *zero-address machine* is a stack-based machine where all operations are done using values stored on the operand stack. For this problem, you may assume that its ISA allows the following operations:
   - PUSH M - pushes the value stored at memory location M onto the operand stack.
   - POP M - pops the operand stack and stores the value into memory location M.
   - OP - Pops two values off the operand stack, performs the binary operation OP on the two values, and pushes the result back onto the operand stack. The popped values are NOT stored back in the memory.

   Note: To compute A - B with a stack machine, the following sequence of operations is necessary: PUSH A, PUSH B, SUB. After the execution of SUB, A and B would no longer be on the stack, but the value of A - B would be at the top of the stack.

2. A *one-address machine* uses an accumulator to perform computations. For this problem, you may assume that its ISA allows the following operations:
    - LOAD M - Loads the value stored at memory location M into the accumulator.
    - STORE M - Stores the value in the accumulator into memory location M.
    - OP M - Performs the binary operation OP on the value stored at memory location M and the value present in the accumulator. The result is stored in the accumulator (ACCUM = ACCUM OP M).
3. A *two-address machine* takes two sources, performs an operation on these sources, and stores the result back into one of the sources. For this problem, you may assume that its ISA allows the following operation:
    - OP M1, M2 - Performs a binary operation OP on the values stored at memory locations M1 and M2 and stores the result back into memory location M1 (M1 = M1 OP M2).
4. A *three-address machine* whose sources and destination are *memory* locations. For this problem, you may assume that its ISA allows the following operation:
    - OP M3, M1, M2 - Performs a binary operation OP on the values stored at memory locations M1 and M2 and stores the result back into memory location M3 (M3 = M1 OP M2).
5. A *three-address machine* whose sources and destination are *registers*. Values are loaded into registers using memory operations. For this problem, you may assume that its ISA allows the following operations:
    - OP R3, R1, R2 - Performs a binary operation OP on the values stored at registers R1 and R2 and stores the result back into register R3 (R3 = R1 OP R2).
    - LOAD R1, M - Loads the value at memory location M into register R1.
    - STORE R2, M - Stores the value in register R2 into memory location M.

To measure memory efficiency, make the following assumptions about all five instruction sets:

- The opcode is always 1 byte (8 bits).
- All register operands are 1 byte (8 bits).
- All memory addresses are 4 bytes (32 bits).
- All data values are 4 bytes (32 bits).
- All instructions are an integral number of bytes in length.

There are no other optimizations to reduce memory traffic, and the variables A, B, C, and D are initially in memory. You are only allowed to use the following instructions: LOAD, STORE, PUSH, POP, ADD, and SUB. As described above, PUSH and POP are only applicable to stack-based machines. For the two-address machines, only ADD and SUB are given to you for answering the questions.

1. Write the code sequences for the following high-level language fragment for each of the five styles. (Optimization is not considered.) Be sure to store the contents of A, B, C, and D back into memory, but do not modify any other values in memory. (30 points, 6 points per instruction set)

```
A = A + C;
B = B - C;
D = B - A;
C = D + A;
```

2. Calculate the instruction bytes fetched and the memory-data bytes transferred (read or written) for each of the five architecture styles. (20 points, 4 points per instruction set)
3. Which architecture is most efficient as measured by code size? (5 points)
4. Which architecture is most efficient as measured by the total memory transfer required? Please consider both code and data. (5 points)

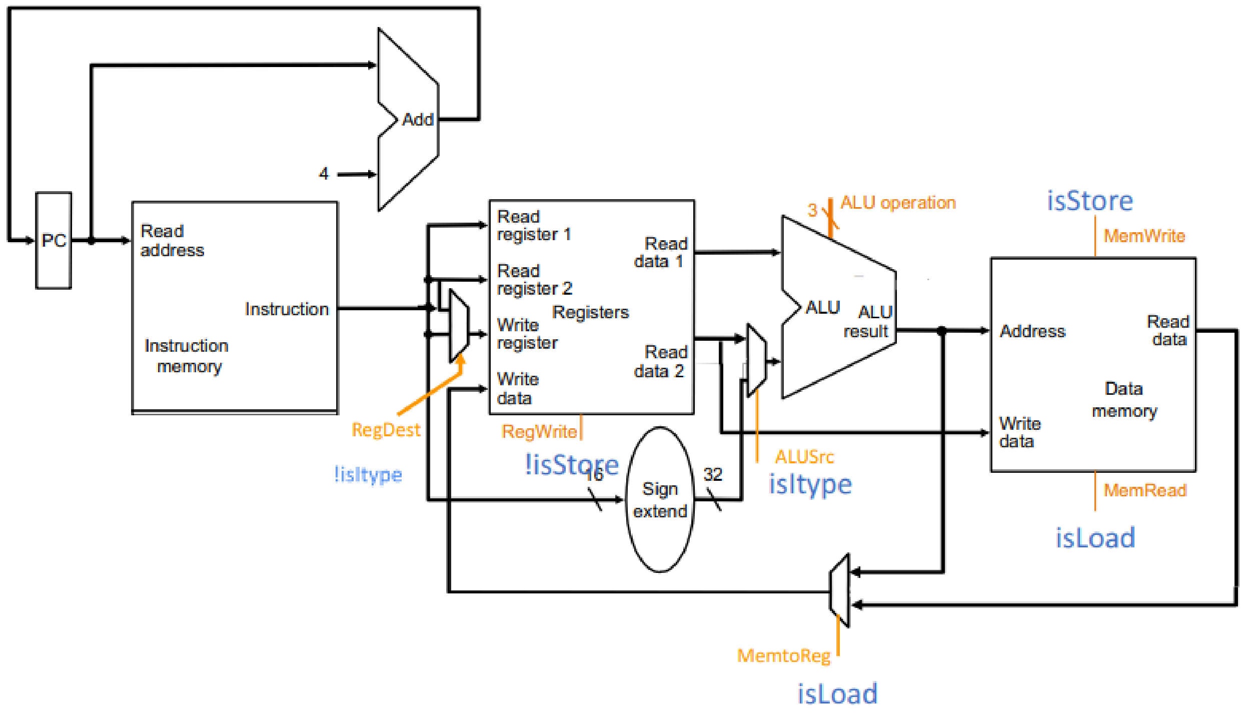
## 3. Single-cycle Processor Datapath (30 points)

In Lecture 5, we briefly discussed the JAL (jump and link) instruction. In this problem, you will modify the single-cycle datapath to complete the datapath and control signals. The JAL instruction has the following semantics:

```
JAL : R31 ← PC+4
PC ← PC31...28 || Immediate || 00
```

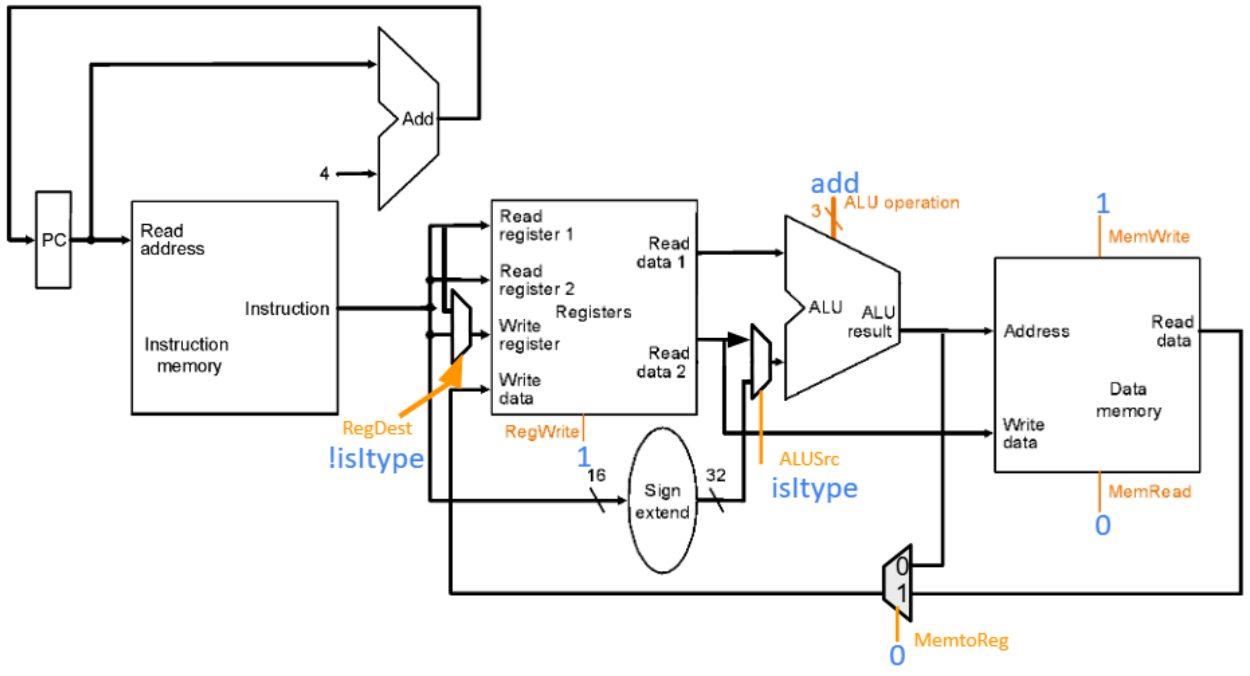
Note: "||" stands for concatenation of bits

Add to the datapath the necessary data and control signals to implement the JAL instruction. Draw and label all components and wires very clearly. Give control signals meaningful names. If selecting a subset of bits from many, specify exactly which bits are selected.



## 4. Instruction Semantics (30 points)

A pesky engineer implemented a mystery instruction on the MIPS single-cycle machine that we are learning in class. Please describe what the instruction does and what the operands are. You may follow the instruction semantics style in Lecture 5.



Add
4
PC
Read address
Instruction
Instruction memory
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
RegDest
!isItype
RegWrite
1
16
Sign extend
32
add
3
ALU operation
ALU
ALU result
ALUSrc
isItype
1
MemWrite
Address
Read data
Data memory
Write data
MemRead
0
0
1
MemtoReg
0