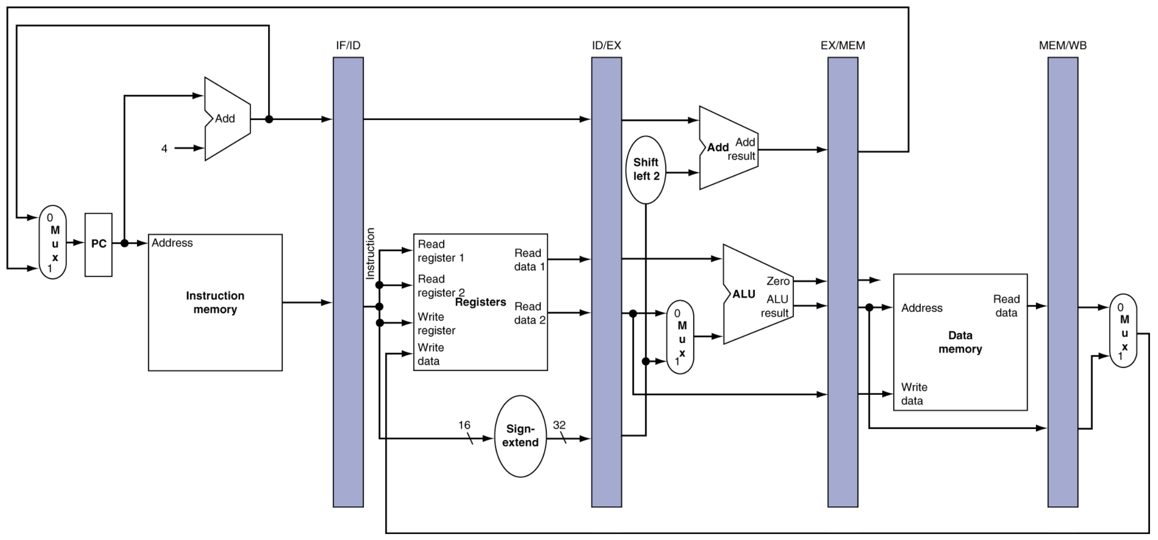

Figure 1: Schematic diagram for a pipelined datapath.

Question 3 (20 points):

1. **(7 points)** Figure 1 is a representation of the architecture of a pipelined datapath for a MIPS processor. However, if we were to implement this pipeline as represented in this schematic diagram, the operation of the processor would be incorrect. Explain why it is incorrect and the required change(s) to the datapath schematics to make its operation correct (this question is about the data flow in the datapath, we assume that the control logic, not shown in the schematic, is correctly implemented).

   The bits that specify which register should be written into the register file is coming from the instruction that is currently in the ID stage, but the value to be written comes from the instruction that is currently in the WB stage. The fix is to pass the bits that specify the register to which an instruction writes (the write-register) along with the instruction execution throughout the pipeline. Then these bits are connected from the MEM/WB register into the register file.

2. (**7 points**) We are writing a MIPS assembly program and we need to load a value from memory and put this value in `$t0`. Then we need to shift this value to the left by 16 bits. Explain why it is a bad idea to place the shift instruction immediately after the load instruction in the assembly program.

   The value loaded from memory is only available at the end of the MEM stage of the pipeline. But it is needed at the started of the EX stage by the shift instructions. If the shift instruction is placed immediately after the load instruction, the needed value is not available even through forwarding and thus an execution cycle is wasted — a bubble appears in the pipeline.

3. (**6 points**) What is the role of dirty bits in a memory hierarchy? Which write policy requires a dirty bit?

   Paraphrasing Alex Michon's answer in the CMPUT 229 class forum (with corrections):

   The dirty bit notifies the system if a change has occurred to the data. The dirty bit is required because if a change has occurred to that block in the cache, the cache controller must first write the modified block into memory before replacing the block. If the dirty bit is 0 then the cache controller can save time by just overwriting the block without worrying about losing changed data. The same can occur for a page with respect to storage in disk.

   A write-back policy requires dirty bits.