University of California at Berkeley
College of Engineering
Department of Electrical Engineering and Computer Science

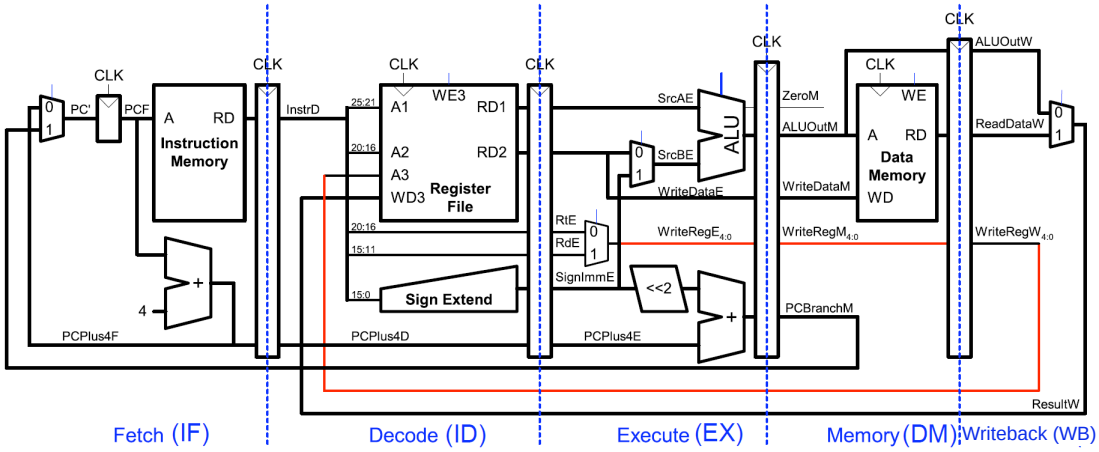
EECS150, Spring 2010

Homework Assignment 6 SOLUTION: Processor Pipeline Design; Composition of Memory Primitives

Due March 5^{th}, 2pm

Homework submissions must be made via the course SVN repository. Email submissions will not be accepted! Please format your homework as plain text, or PDF. You may use PNG for any necessary figures. Use a CAD program for diagrams (Microsoft Visio is installed on the machines in 125 Cory).

1. Identify all hazards in the datapath shown here, asuuming the instruction set contains only the following MIPS instructions:



`ISA: addu, addiu, beq, j, jr, jal, lw, sw`

***Remember**: you learned about structural, control, and data hazards.*

(a) Structural

- This datapath does not have the hardware needed to implement j-type instructions.

(b) Control

- Branches are resolved in the EX stage, resulting in delayed branching (by 3 cycles).

(c) Data

- This datapath does not have the hardware needed to implement j-type instructions.

2. Extend the datapath from problem 1) to add the following I-type instructions (assuming accesses into memory must be word-aligned, meaning the lowest 2 bits of address are ignored, and operate on 32-bit words):

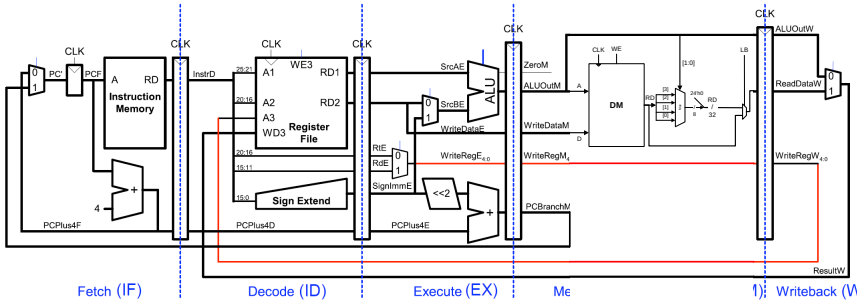
(a) **`lb`** : load byte : (loads the value `32'h000000??` according to:

$$R[\$rt] = M[(R[\$rs] + SignExt(imm))]$$

The address specified need not be word-aligned (but the hardware must implement this using word-aligned accesses!).

Since you will be drawing a lot of datapaths for your design documents this semester, take the time to get quick at it, and to build up a library of standard diagram components (you should be using a CAD tool).

Since only a word addressed memory is available, we need a means to align the loaded word to the desired byte boundary, and to select one of the 4 bytes loaded. In other words, we need to shift and mask the loaded bytes.



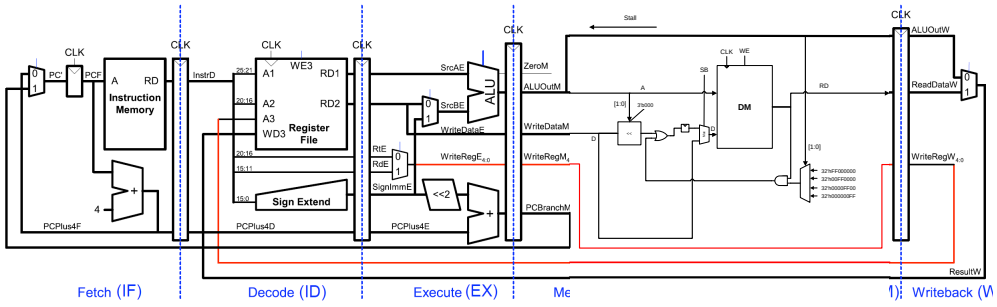
(b) **`sb`** : store byte : (modifies a single byte in memory, leaving the other 3 bytes of the corresponding word unchanged. The instruction semantics:

$$M[(R[\$rs] + SignExt(imm))] = R[\$rt]$$

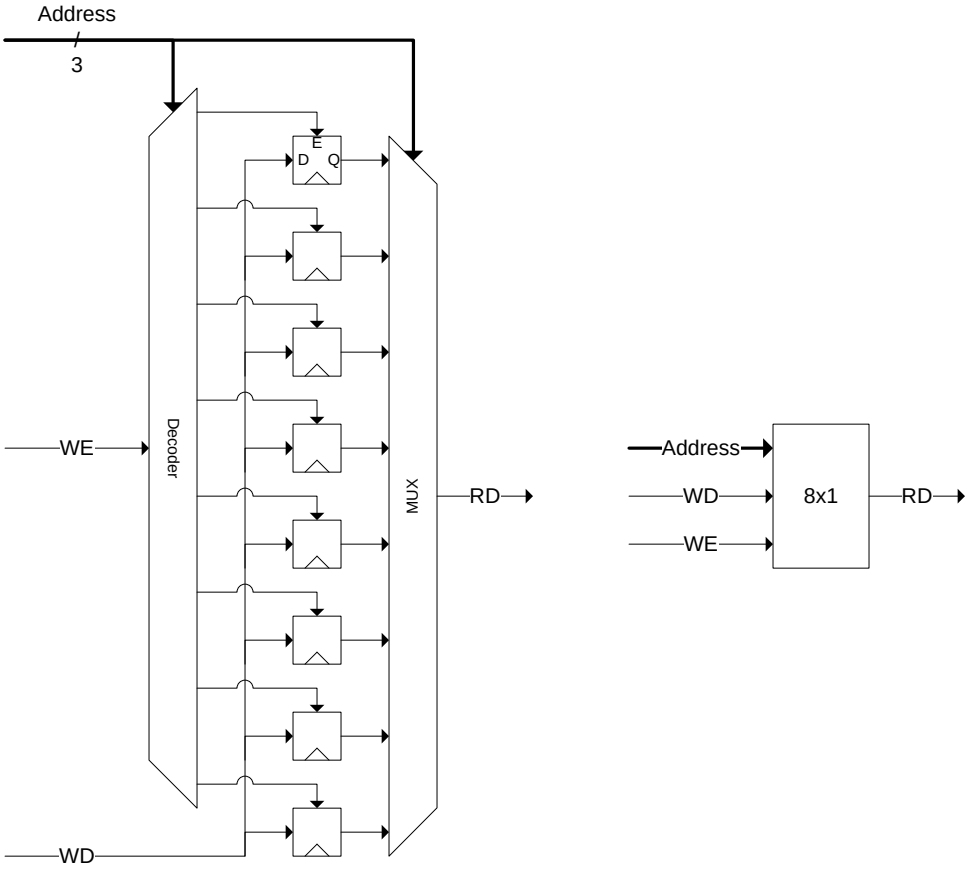
Again, the address specified need not be word-aligned.

Hint*: you will need to perform a read-modify-write sequence to do this correctly. Can this be done while maintaining a CPI of 1?*

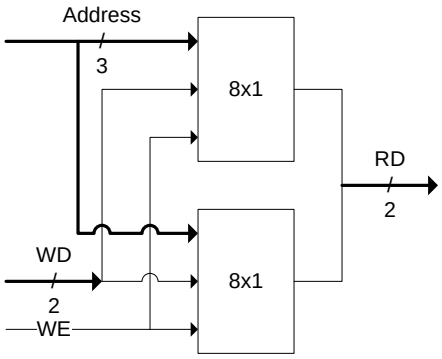
The granularity of memory access provided by our datapath is 4 word-alinged bytes, which is less precise than what is needed to store single bytes. For this reason, a read-modify-write sequence is required to complete a byte-aligned store. Two memory accesses are needed for the load byte instruction, so CPI can no longer be 1 (this instruction needs 2 memory accesses, and thus must stall the pipeline for at least 1 cycle).



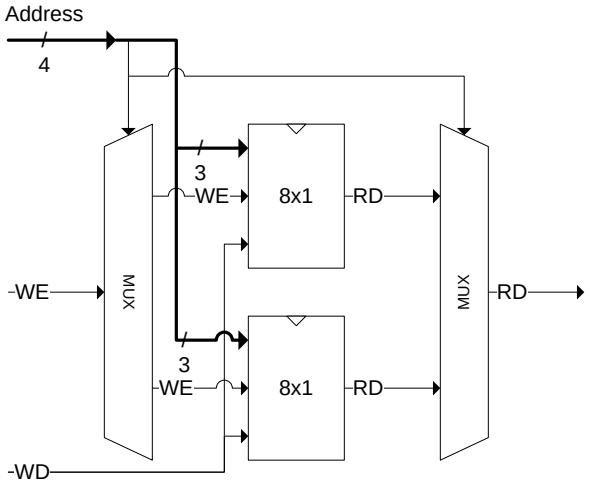
3. Consider the following model of a single-port 8x1 memory.



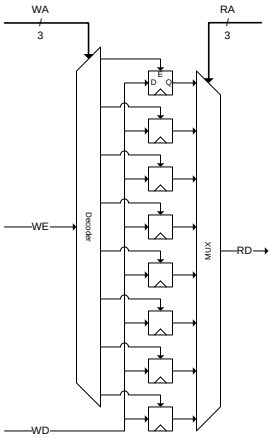
(a) Using one or more such memories (unmodified), construct a 8x2 memory.



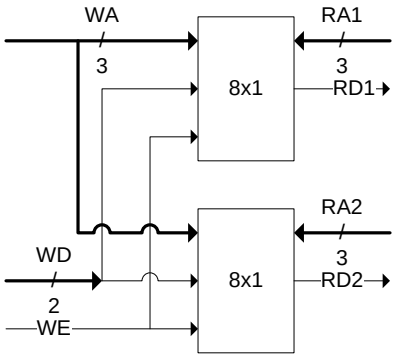
(b) Using one or more such memories (unmodified), construct a 16x1 memory.



(c) Modify the model to describe an synchronous single-write, asynchonous single-read 8x1 memory (two address inputs).

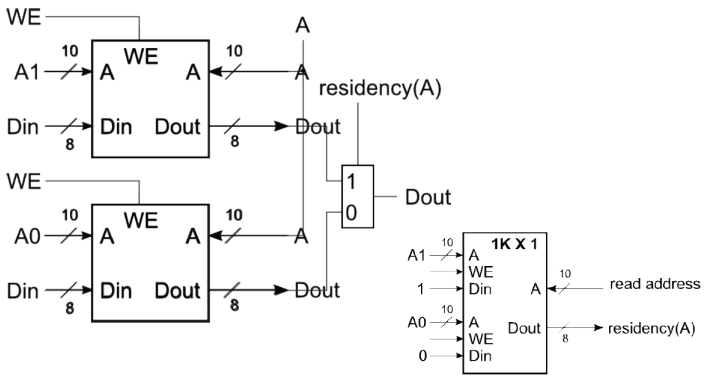


(d) Using one or more memories from c), construct a 8x1 memory with two read ports and one write port.



(e) Using one or more memories from c), construct a 8x1 memory with two write ports and one read port.

This cannot be efficiently done using the provided memory primitives, as the residency table lookup trick would cost an additional 8x1 dual write, dual read memory, which is a more complex problem than the one we are trying to solve.



4. **DDCA: 7.25**

Use the 5-stage pipelined processor from lecture 11 to answer the question. To avoid drawing unnecesary detail, you may use a format such as slide 6 of lecture 12.