CS 151B: Homework 3

Jonathan Woong

804205763

Spring 2017

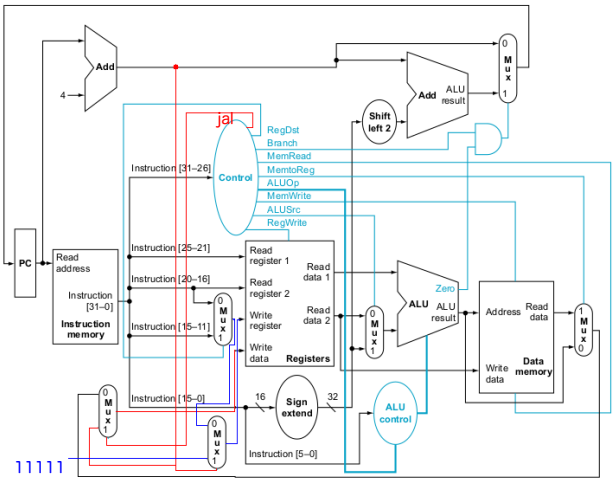
Discussion 1A

Wednesday 26$^{\text{th}}$ April, 2017

Problem 1

Consider the single cycle MIPS implementation with the datapath shown in Figure 4.17 (page 265) in the book and the control logic implemented based on the truth table in Figure 4.22 (page 269). Your task is to modify this implementation to add support for the `jal` (jump and link) instruction, which is part of the MIPS ISA. This instruction is described on page A-63 in the book.

A) Show any required modifications to the datapath. Also explain the modifications in words. In addition, if you need to modify any of the datapath building blocks, draw the implementation of the modified building blocks and explain the modifications in full detail.



The value PC+4 and original Write Data are inputs to a MUX that is controlled by the `jal` signal. If the `jal` signal is 1, the PC+4 value will be written to the Write register, and if it is 0 the original Write Data will be written to the Write Register. The output of the MUX controlled by the RegDst signal is the first input to a new MUX. This new MUX also takes as input the value 11111 representing register $ra. The `jal` signal also controls this MUX, and if it is 1, the MUX will output 11111 as the input to Write Register. If it is 0, the MUX will output what the MUX output controlled by the RegDst signal.

B) Are any new control signals required? If so, list them with an explanation and identify them on the datapath diagram.

Yes, the `jal` signal is required.

C) Show any necessary changes to the control logic by presenting a modified version of the control truth table in Figure 4.22. For any entries that you add, mark all "don't cares" with an X.

opcode	jal	RegDst	MemRead	MemtoReg	ALUOp	MemWrite	ALUSrc	RegWrite
000110	1	X	0	X	XX	0	X	1

Problem 2

Write a recursive procedure in MIPS assembly language to compute the following function:

$$rfunc(n) = \begin{cases} 3 & \text{if } n \leq 0 \\ 2 & \text{if } n = 1 \\ rfunc(\lfloor \frac{n}{4} \rfloor + 1) + rfunc(\lfloor \frac{n}{8} \rfloor - 1) & \text{if } n > 1 \end{cases}$$

Note that divides by powers of 2 can be implemented using shifts. The argument n is in register \$a0 and the result $rfunc(n)$ should be returned in register \$v0. Your implementation should follow the definition above in a straightforward way: for $n > 1$, your code should recursively call $rfunc(\lfloor \frac{n}{4} \rfloor + 1)$ as well as $rfunc(\lfloor \frac{n}{8} \rfloor - 1)$. Within this constraint, make your code as efficient as possible.

```
rfunc:  addi    $29, $29, -12    # adjust sp for 3 items
        srl     $8, $4, 2        # t0 = n/4
        addi    $8, $8, 1        # t0 = n/4 + 1
        sw      $8, 0($29)       # save t0 to stack
        srl     $9, $4, 3        # t1 = n/8
        addi    $9, $9, -1       # t1 = n/8 - 1
        sw      $9, 4($29)       # save t1 to stack
        sw      $31, 8($29)      # save return address
        ble     $4, $0, L1       # branch to L1 if n<=0
        li      $10,1            # t2 = 1
        beq     $4, $10, L2      # branch to L2 if n=1
        slt     $11, $10, $4     # t3 = 1 if n > 1
        beq     $11, $10, L3     # branch to L3 if n > 1
        addi    $29, $29, 12     # pop 3 items off stack
        jr      $31              # return to caller

L1:     li      $2, 3    # v0 = 3
        jr      $31      # return to caller

L2:     li      $2, 2    # v0 = 2
        jr      $31      # return to caller

L3:     add     $4, $8, $0       # a0 = n/4 + 1
        jal     rfunc            # call rfunc with (n/4 + 1)
```

```
add     $16, $2, $0     # s0 = rfunc(n/4 + 1)
add     $4$, $9, $0     # a0 = n/8 - 1
jal     rfunc           # call rfunc with (n/8 - 1)
add     $17, $2, $0     # s1 = rfunc(n/8 - 1)
add     $2, $16, $17    # return rfunc(n/4 + 1) + rfunc(n/8 - 1)
jr      $31             # return to caller
```

Problem 3

Consider the single cycle MIPS implementation with the datapath shown in Figure 4.17 (page 265) in the book and the control logic implemented based on the truth table in Figure 4.22 (page 269). The implementation of the ALU in the datapath is shown in Figure B.5.12, page B-36. Assume that in the original implementation of the ALU, the time (latency) to perform an AND operation is exactly the same as the time to perform an OR operation. Due to a change in the implementation, the time to perform an AND operation is doubled. We would like to predict how this change will affect the time it will take the processor to execute a program that includes many AND instructions and many OR instructions. The comparison is for the same program executing on the original implementation and the new implementation.

A) Is it possible that the change in the ALU will result in a change in program execution time?
   Yes.

B) Explain in detail your answer to Part A.
   Execution time is defined as the time between the start and completion of a task. Doubling the time it takes to complete AND operations in a program with many AND operations can possibly increase the amount of time between the start and end of the program.

C) Is it likely that the change in the ALU will result in a change in program execution time?
   Yes.

D) Explain in detail your answer to Part C.
   Due to the fact that the program consists of many AND operations, the change to the ALU will more likely increase the amount of time needed for the program to start and finish execution when compared to a program with few AND instructions.

Problem 4

Consider the multicycle MIPS implementation shown in the 3rd edition of the book, Figure 5.28 (page 323) and Figure 5.37 (page 338). Due to a hardware fault (malfunction), the MemtoReg output from the control unit is always 0 (stuck-at-0). Explain in full detail what will be the consequences of this change when the processor executes programs – how will it change the behavior of the processor as observed by a user/programmer who does not know and does not care how the processor is implemented internally? Be sure to clearly identify each and every consequence of this fault in as much detail as possible.

At the circuit level, if the MemtoReg output is stuck-at-0, then the output of the MUX that takes that signal will never output the output of Memory data register. The only output of the MUX will be the output from ALUOut.
At the programmer level, instructions that require loading a word from memory to register will not behave as expected. This means that the programmer cannot write a program that performs arithmetic on data that comes from memory. Instead, any data that the programmer wants to manipulate must already be in the registers or must be derived through performing arithmetic the values already in registers.

Problem 5

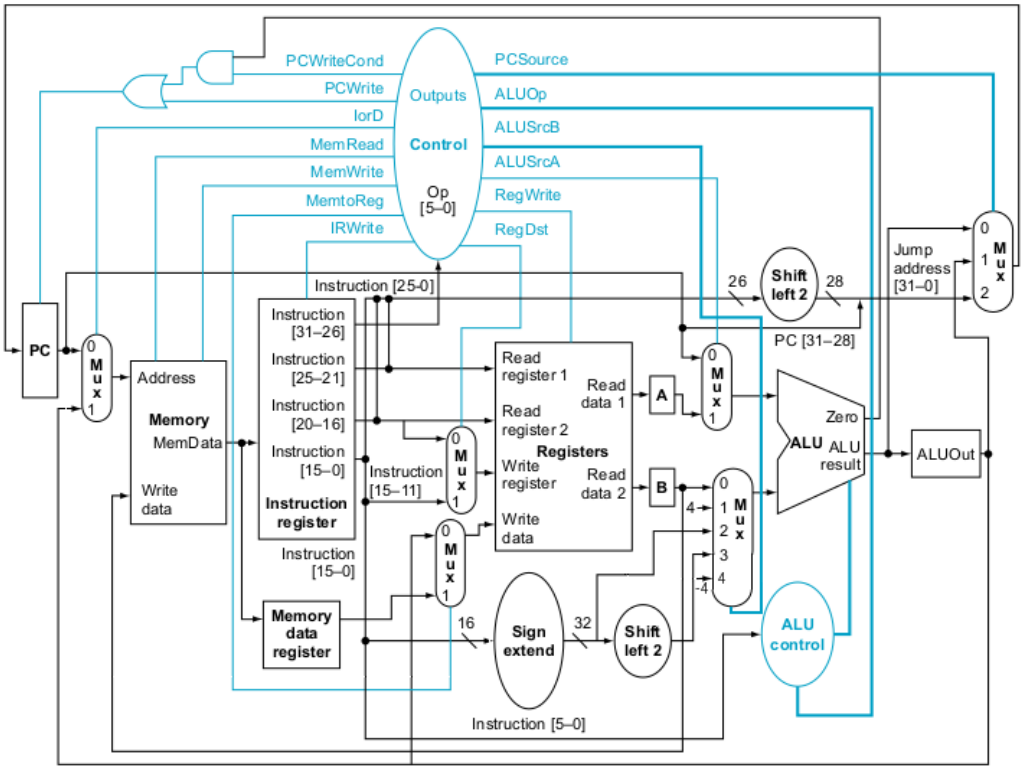
Consider the multicycle MIPS implementation shown in the 3rd edition of the book, Figure 5.28 (page 323) and Figure 5.37 (page 338). Your task is to modify this implementation so that it supports a new instruction, `swpc` (store word PC). The format of this instruction is exactly the same as the format of the `sw` instruction, except that the opcode is 010111. The `swpc` instruction stores in memory the address from which the instruction is fetched. Thus, for example, if the `swpc` is at address 180, it will store in memory the 32-bit number 180. The address into which the PC value is stored is computed in exactly the same way as the destination for the `sw` instruction. You cannot modify the internal implementation of any of the datapath modules. The only exception is that you can replace an existing MUX with a MUX that has more inputs (if needed, you must add the required additional control signals). You must note such a replacement of the MUX when describing changes to the datapath. You cannot add any ALUs, adders, or subtractors to the datapath. Your first priority is not to slow down any of the existing instructions. Your second priority is to minimize the changes to the datapath. Your third priority is to minimize the execution time of the new `swpc` instruction. Your fourth priority is to minimize the changes to the control. Note that you cannot speed up any of the building blocks used to construct the implementation in Figure 5.28. Furthermore, all the existing functionality must be maintained.

A) Explain the basic idea of your modifications in 2-4 clear sentences.

   I added one more input to the MUX controlled by ALUSrcB. The input is the value -4. Using the current funcitionality of the MIPS implementation, we can unincrement the PC by 4 to obtain the PC value before it was incremented. Then, we use the current functionality in the `sw` instruction to store the unincremented PC into the calculated memory address.

B) Show the necessary modifications to the datapath by showing them on a copy of Figure 5.28 (from 3rd

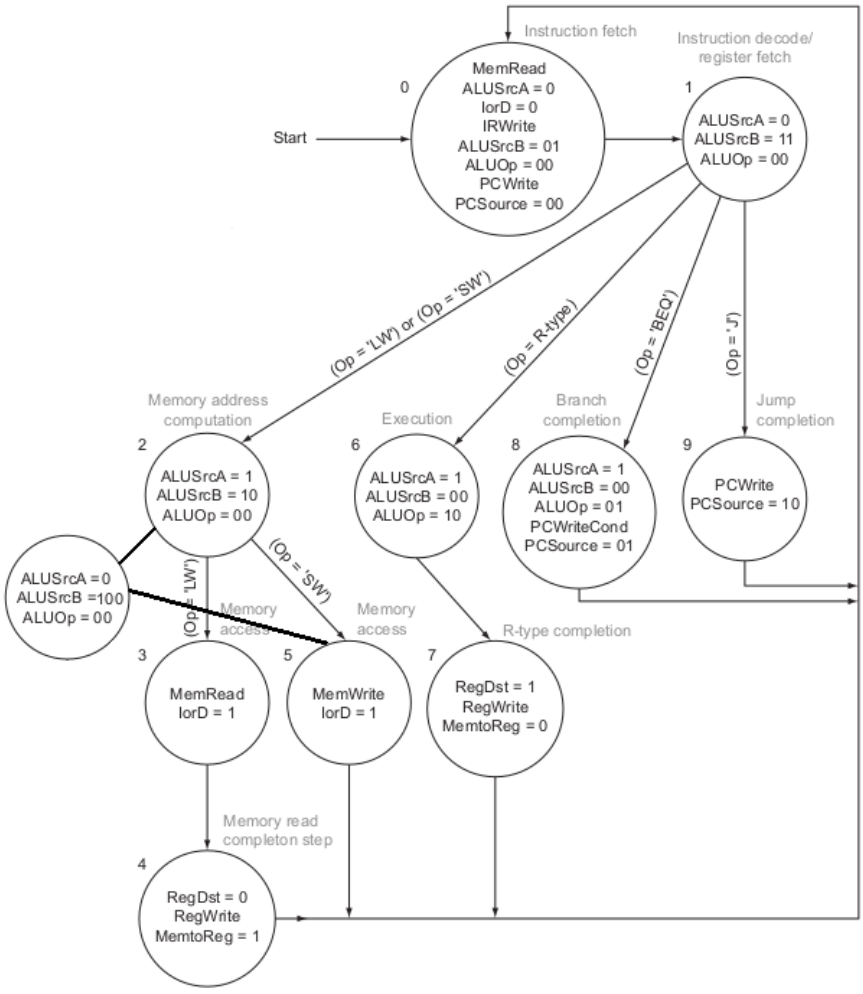
edition)



C) Are any new control signals required? If so, list them with an explanation and identify them on the datapath diagram.

No new control signals are required, but ALUSrcB must now output a 3-bit signal rather than a 2-bit signal. The new signal can be 100.

D) Modify the control unit state diagram (Figure 5.37 from 3rd edition) to reflect the added support for the `swpc` instruction.



E) With your implementation, how many cycles does it take to execute the new instruction?

5 cycles.