CS 2110 Lab 7
Intro into LC-3

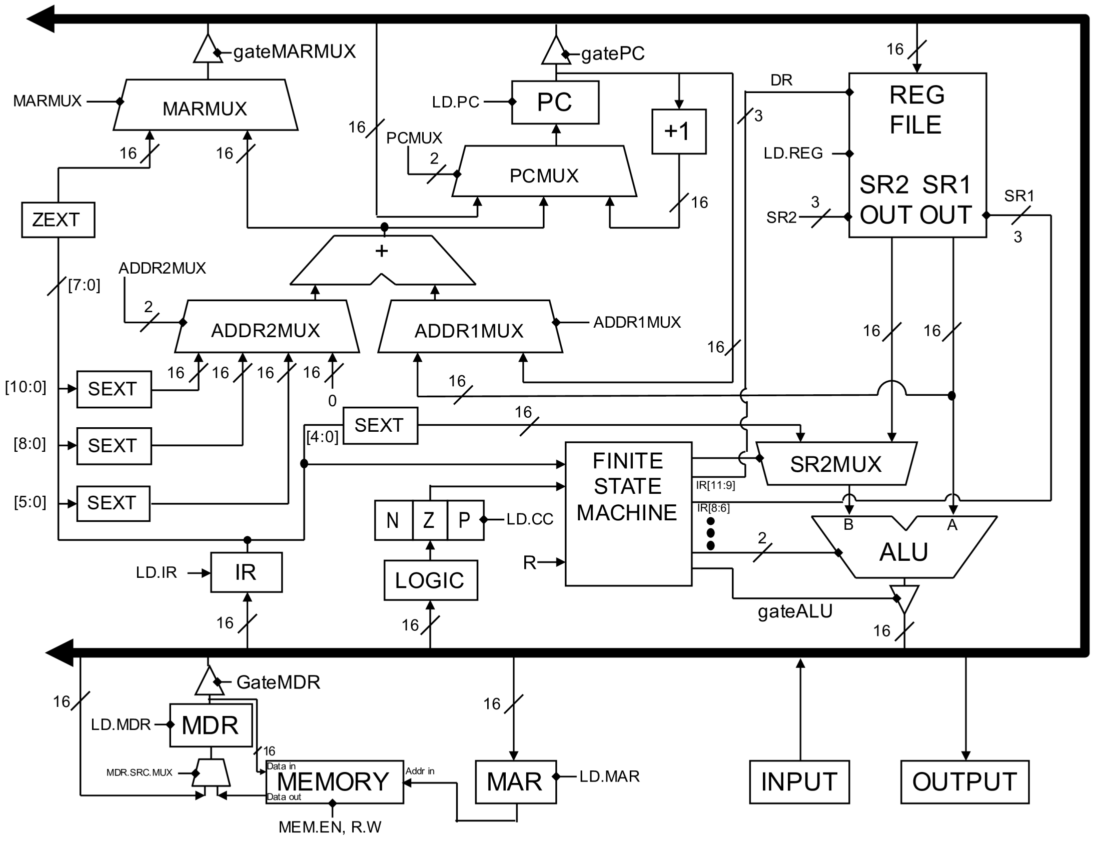
The TAs :)

09/17/2018

1 Recitation Outline

1. LC-3 Components:
   - The Bus: Wire transferring data between components on the datapath
   - Clock Cycles: Taking steps to prevent short circuit conflicts
   - The Micro-controller: A finite state machine. Each of the different states determines what control signals will be asserted on the datapath.
   - Memory: Loading/Storing data in memory
   - The PC: The PC that holds the current program counter.
   - The ALU: The logical unit that performs computations.
2. Tracing Example

2 Components of the LC-3



2.1 The Bus

The bus is a 16-bit wire on the datapath that transfers data between many components. The biggest concept with this element is that only one piece of data can be on the bus at a time. In order to ensure this, there are tri-state buffer gates (such as GateALU, GateMARMUX, etc) that will allow data to be driven to the bus only when the flag is set.

Example: A hallway with lots of doors leading to different classrooms. In this hypothetical situation, this hallway only allows one person in it at a time. The hallway is analogous to the bus, only allowing one piece of data to slide through at a time. The doors leading into the classrooms represent the LD write enables, allowing data to enter a sub component. The doors leading out of the classrooms represent the tri-state buffer gates that allow output from the sub component onto the bus/hallway.

2.2 Clock Cycles

In order to prevent short circuit conflicts (i.e. having two different values on the same wire), the LC-3 runs on clock cycles (alternating 0/1's). Clock cycles, when handled correctly, ensure that we don't run into short circuits by making sure that every wire and component only hold one value at a time. This is why we have the tri-state buffers before any value writes to the bus, if two different components were to write to the bus at the same time, we would have a short-circuit in the bus. Instead, we can gate any writing to the bus so that we can ensure that only one component writes to the bus at any given moment in time.

2.3 The Micro-controller

The micro-controller is a finite state machine. Similarly to your state machine homework, the micro-controller switches in between states. Each of these states determine what flags need to be set in the datapath. In Homework 4, the outputs of your state machine (A, B, C) controlled the gate motors or the "Scan Card Now" light. Here, the output of the current micro-controller state determines what control flags will be set in the datapath, ensuring that correct operations take place, while preventing any short circuits.

2.4 Memory

- On a load: (Loads a value at a certain memory location onto the bus)
  - Load the address of the data in the Memory Address Register (MAR)
  - Memory is read and the data at that memory is delivered to the MDR
- On a store: (Stores a value from the bus at a certain memory location)
  - The data is placed in the Memory Data Register (MDR)
  - The address in which the data will need to be stored is placed in the MAR
  - if the proper signals are asserted, the data will be inserted into memory at that proper address

2.5 PC

The PC holds the current program counter — the address in memory of the next instruction to execute. In FETCH, the value held in the PC register is driven onto the bus and into the MAR register. Memory is then read at that address and the data from this memory access is stored in the IR register.
BIG NOTE: The PC holds the address of the *next* instruction, the IR holds the current instruction itself.
Three ways that PC gets updated:

- PC + Offset
  Due to a branch instruction, the incremented PC will be added to an offset to generate the new PC.
  Hint: At some point, you will need to calculate the offset. An easy way to do this is:

  branching address - incremented PC = offset

- PC + 1
  The general format for the PC being updated - it increments by one.
- Given a value for the PC via the bus. This will not be used frequently.

2.6 ALU

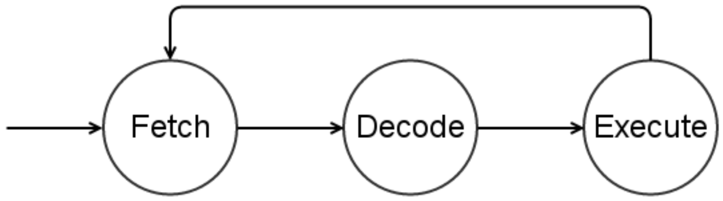
The ALU is very similar to the ALU you created in Homework 3. It has two inputs:

- One of the inputs will be from data from the register file
- The other input could either be data from the register file or an offset generated from the IR.

The ALU will drive the output onto the bus. As well, with a certain flag set, this output will set the condition codes - N, Z, P, based on the value of the output.

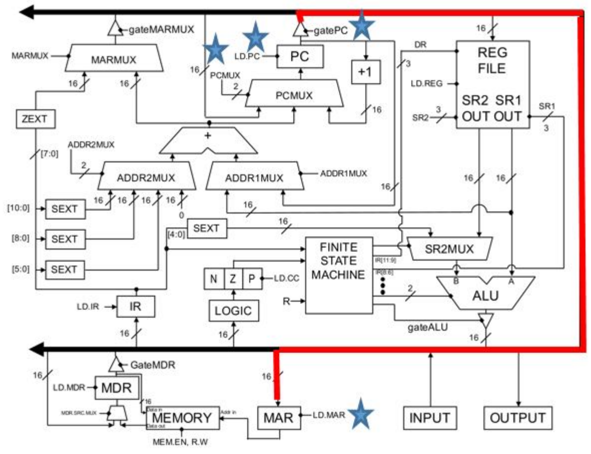
3 Tracing Example

`ADD R3 R2 5`

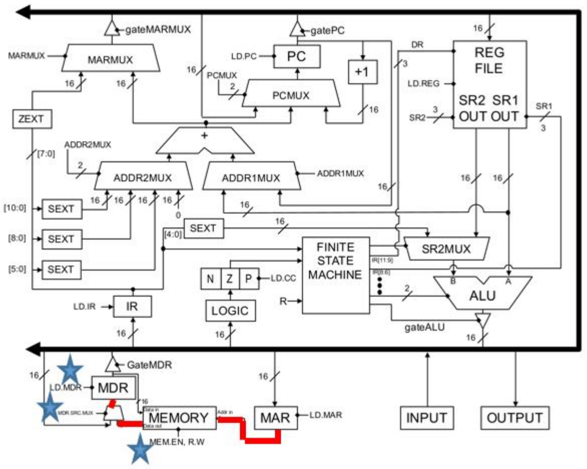


3.1 Fetch

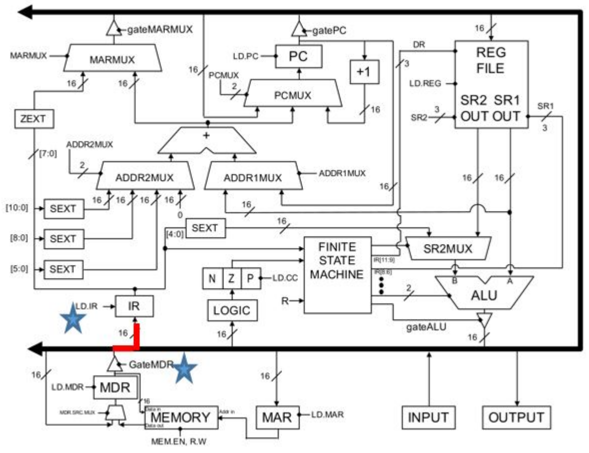
- Assert GatePC to drive the PC value onto the bus. Turn on LD.MAR to load this value into the MAR register. At the same time, the value in the PC will be incremented, by selecting that line in the PCMux and loading PC.



- Assert the MEM.EN and LD.MDR bits to enable memory to be read and the data at this address to be loaded into MDR. Also set the MDR.SRC.MUX so the input to the MDR is from Memory, not the bus.



- The contents of MDR loaded into the IR



3.2 Decode

The contents of the IR are decoded, leading to the control logic providing the correct control signals.

3.3 ADD

- The ALU grabs the register value depicted from SR1 from the register file (the A input). The offset from the IR is sign extended and loaded as the B input. The specific operation is performed in the ALU and GateAlu opens, allowing the data to flow onto the bus. At this point, the Ld.CC flag is turned on, setting the condition code. The Ld.Reg flag is turned on and the register specified by DR will be loaded with this information.