Lecture 2

Control Unit Design

High-level view of the LC-3b processor

As its name implies, the control unit controls the operation of the data path by orchestrating the operation of the data path. This is accomplished by asserting and de-asserting the appropriate control signals in each clock cycle. All meaningful digital systems (and not just microprocessors) require control units, therefore understanding the basic operation and design of control units is essential for anyone who designs digital systems.

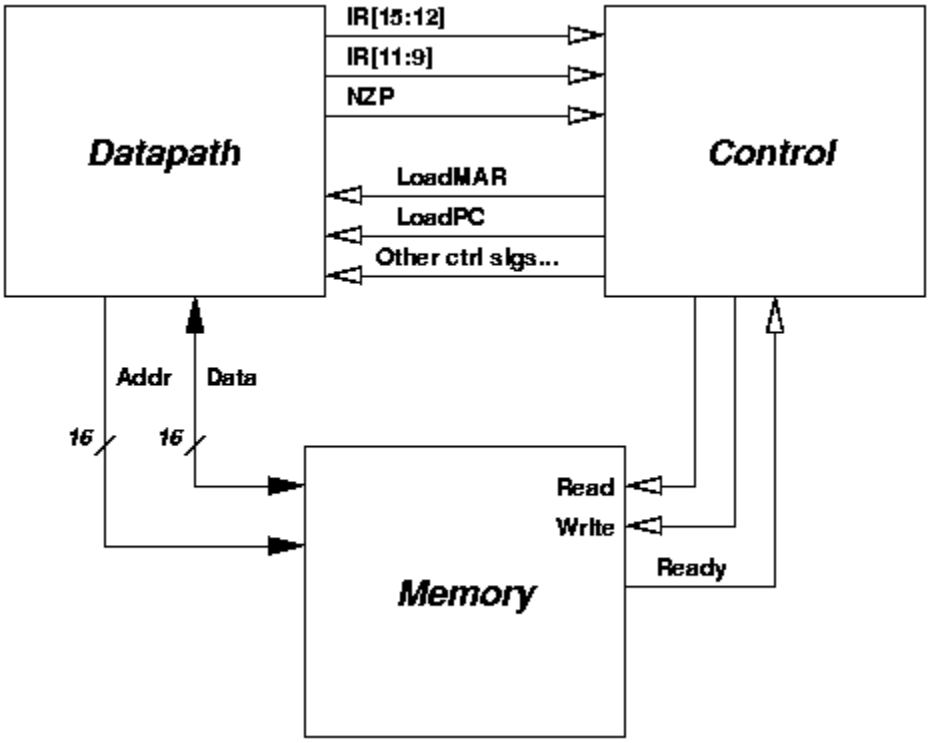

Figure 1 High-level view of control-unit in a LC-3b system

Figure 1 depicts the interface between control unit and data path of an LC-3b system. Communication between the control unit and the data path is accomplished with two types of signals between them. Signals that travel from data path to the control unit are referred to as **condition** signals. These signals deliver relevant data path state information that the control unit needs in order to make state transition decisions. Signals that travel from the control unit to the data path are called control signals. These signals dictate the actions to be performed by the data path. Some of the control signals can go from the control unit to an external component. For example, the read/write signals go to the

memory in Figure 1. Similarly, some of the condition signals can be go from an external component (memory) to the control unit, such as the ready (memory response) signal.

We assume that at the time of designing control units, the register transfer program and the data path for the desired function(s) have already been derived. The data path for LC2 is depicted in Figure 1. This is indeed a common situation in the real world: the control units are designed given a register transfer program and a data path. The control unit realizes the control transfer statements in the register transfer program. In each cycle, the control generates the data path control signals to execute the corresponding register transfer statement.

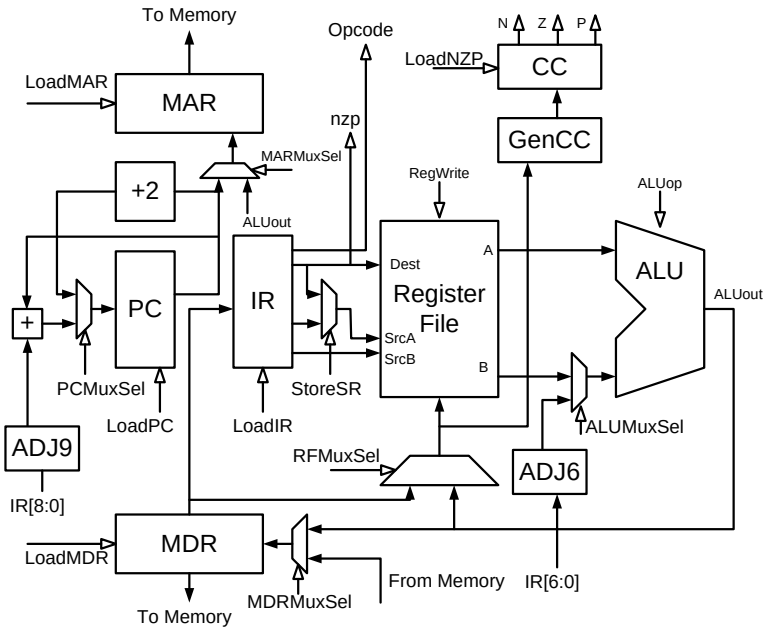

Figure 2 Review of LC-3b data path

Control Unit Design

The first step in designing a control unit is to construct a state diagram according to the given register transfer program. A register transfer instruction consists of a group of statements to be executed in one cycle. Each instruction in the register transfer program directly corresponds to one node in the state diagram. Each node is labeled with the control signal activated by the instruction. Control transfers, implicit and explicit, correspond to arcs in the state diagram. Each arc is labeled with the corresponding condition signal specified in the register transfer program. This process is best illustrated with an example where the LC-3b register transfer program in Figure 3 is translated into a control flow graph.

```
ADD:    MAR <- PC;   PC <- PC + 2;       // Fetch
AND:    MDR <- Mem[MAR];                 // Repeat until memory is ready
```

```
NOT:    IR  <- MDR;
        Rd  <- Ra op Rb;                // op derived from opcode

LD:     MAR <- PC;  PC <- PC + 2;       // Fetch
        MDR <- M[MAR];                  // Repeat until memory is ready
        IR  <- MDR;
        MAR <- PC[15:12] || IR[8:0];    // Calc effective address
        MDR <- M[MAR];                  // Repeat until memory is ready
        Rd  <- MDR;

ST:     MAR <- PC;  PC <- PC + 2;       // Fetch
        MDR <- M[MAR];                  // Repeat until memory is ready
        IR  <- MDR;
        MAR <- PC[15:12] || IR[8:0];    // Calc effective address
        MDR <- Ra;
        M[MAR] <- MDR;                  // Repeat until memory is ready

BR:     MAR <- PC;  PC <- PC + 2;       // Fetch
        MDR <- M[MAR];                  // Repeat until memory is ready
        IR  <- MDR;
        if (cond)
           PC <- PC[15:12] || IR[8:0];
```

Figure 3. Register Transfer Language Description of LC-3b

The state diagram is shown in Figure 4. Note that the state diagam can be viewed as a simplified register transfer program where the register transfer statements are implicitly represented by the nodes and the control transfers are highlighted. This simplification allows the control unit designers to focus on the functions of the control unit and ignore the details of the data path. As long as the control unit activates the control signals correctly at each cycle, the data path design guarantees that the register transfer statements will be faithfully executed.

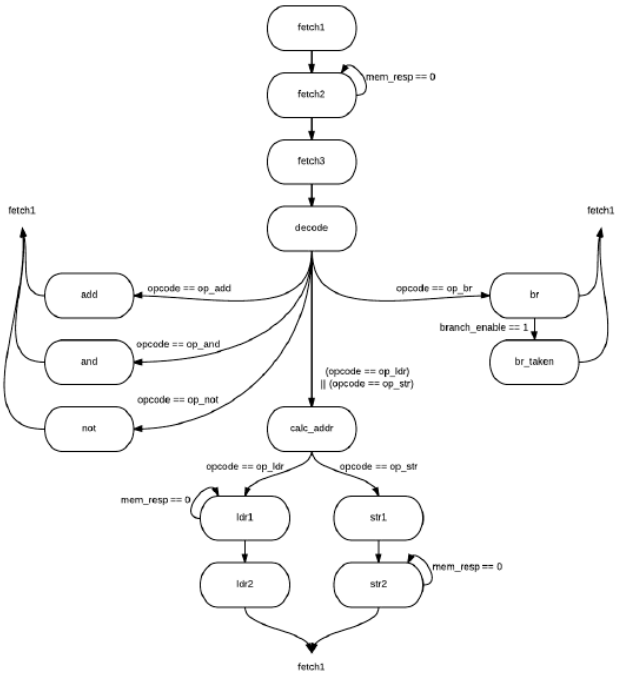

Figure 4. State Diagram

The second step of control unit design is to realize a finite state machine whose states correspond to the nodes in the state diagram. The state transitions in the finite state machine correspond to the arcs. Also, the finite state machine must activate the control signals correctly in each state. Some tools take a high-level description of control logic (e.g., SystemVerilog) and synthesize the necessary control logic to accomplish that state machine. You are using the latter in ECE411.

A finite state machine is realized with registers, combinational logic, and wires. Once constructed, design can be changed only through physically rewiring the unit. Therefore, the resulting circuits are called hardwired control units.

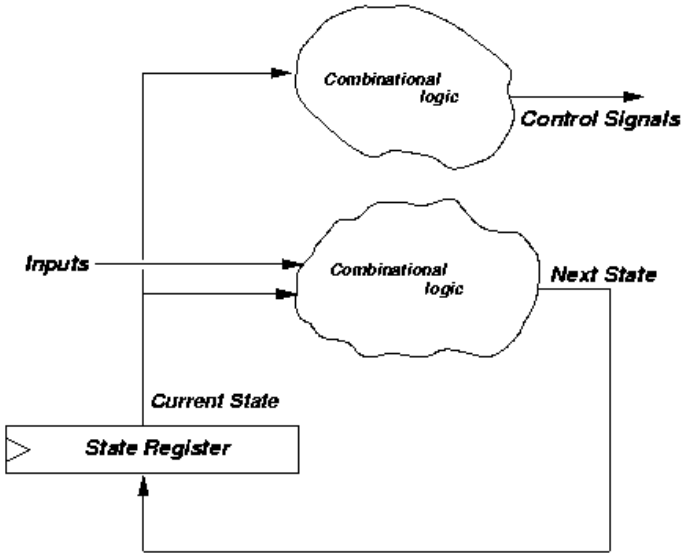

Figure 4. Basic concept of hardwired control

The general structure of a finite state machine controller is given in Figure 4. Two blocks of combinational logic need to be created: One to generate the next state, and one to generate the control signals that control the data path. In ECE411, you will describe the two combinational logic blocks using SystemVerilog and use a tool to generate the actual design.