# Appendix C

# The Microarchitecture of the LC-3b, Basic Machine

This appendix illustrates one example of a microarchitecture that implements the base machine of the LC-3b ISA. We have not included exception handling, interrupt processing, or virtual memory. We have used a very straightforward non-pipelined version. Interrupts, exceptions, virtual memory, pipelining, they will all come later – before we part company in December.

## C.1 Overview

Figure C.1 shows the two main components of an ISA: the *data path*, which contains all the components that actually process the instructions, and the *control*, which contains all the components that generate the set of control signals that are needed to control the processing at each instant of time.

We say, "at each instant of time," but we really mean: *during each clock cycle*. That is, time is divided into **clock cycles**. The cycle time of a microprocessor is the duration of a clock cycle. A common cycle time for a microprocessor today is 0.67 nanoseconds, which corresponds to 1 1/2 billion clock cycles each second. We say that such a microprocessor is operating at a frequency of 1.5 Gigahertz.

At each instant of time—or, rather, during each clock cycle—the 35 control signals (as shown in Figure C.1) control both the processing in the data path and the generation of the control signals for the next clock cycle. Processing in the data path is controlled by 26 bits, and the generation of the control signals for the next clock cycle is controlled by nine bits.

Note that the hardware that determines which control signals are needed each clock cycle does not operate in a vacuum. On the contrary, the control signals needed in the "next" clock cycle depend on all of the following:

1. What is going on in the current clock cycle.
2. The LC-3b instruction that is being executed.

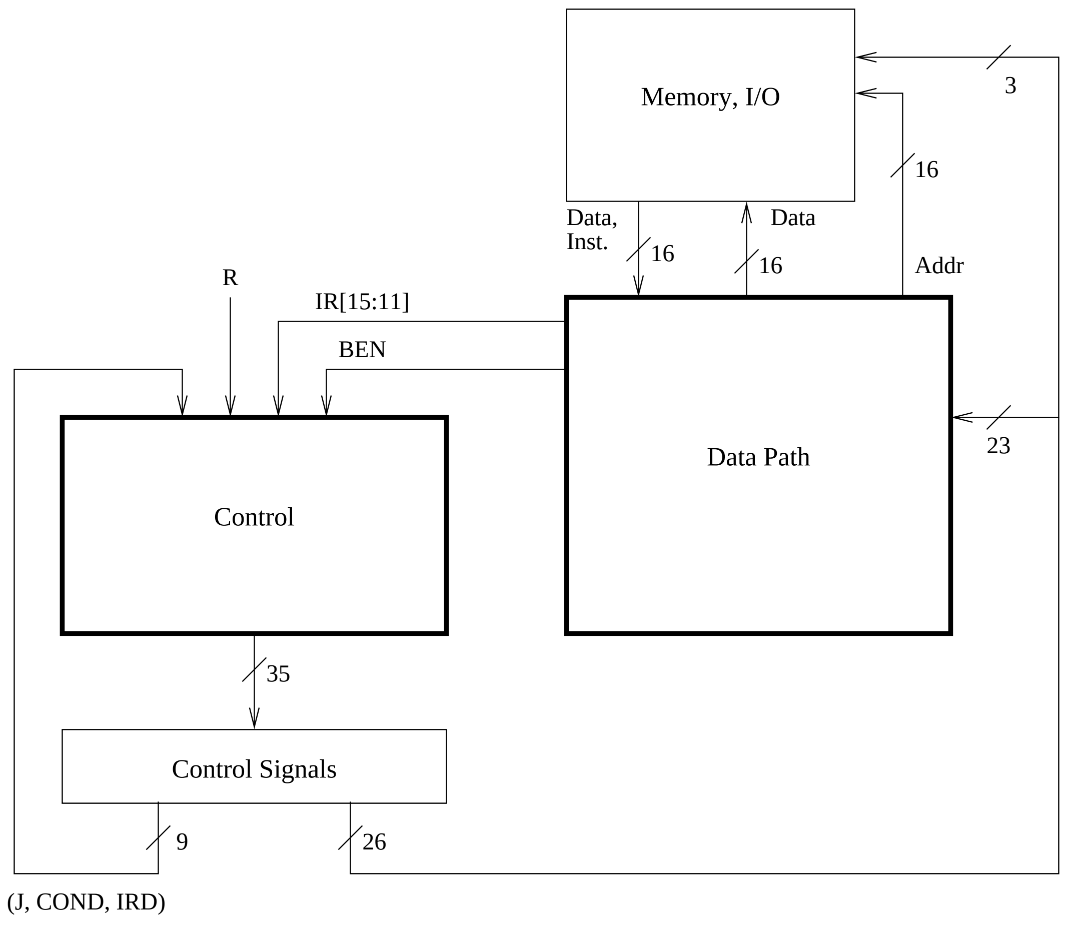

Figure C.1: Microarchitecture of the LC-3b, major components

3. If that LC-3b instruction is a BR, whether the conditions for the branch have been met (i.e., the state of the relevant condition codes).

4. If a memory operation is in progress, whether it is completing during this cycle.

Figure C.1 identifies the specific information in our implementation of the LC-3b that corresponds to these five items. They are, respectively:

1. J[5:0], COND[1:0], and IRD—9 bits of control signals provided by the current clock cycle.

2. inst[15:12], which identifies the opcode, and inst[11:11], which differentiates JSR from JSRR (i.e., the addressing mode for the target of the subroutine call.

3. BEN to indicate whether or not a BR should be taken.

4. R to indicate the end of a memory operation.

## C.2 The State Machine

The behavior of the LC-3b microarchitecture during a given clock cycle is completely determined by the 35 control signals, combined with seven bits of additional information (inst[15:11], BEN, and R), as shown in Figure C.1. We have said that during each clock cycle, 26 of these control signals determine the processing of information in the data path and the other 9 control signals combine with the seven bits of additional information to determine which set of control signals will be required in the next clock cycle.

We say that these 35 control signals specify the *state* of the control structure of the LC-3b microarchitecture. We can completely describe the behavior of the LC-3b microarchitecture by means of a directed graph that consists of nodes (one corresponding to each state) and arcs (showing the flow from each state to the one(s) it goes to next). We call such a graph a *state machine*.

Figure C.2 is the state machine for our implementation of the LC-3b. The state machine describes what happens during each clock cycle in which the computer is running. Each state is active for exactly one clock cycle before control passes to the next state. The state machine shows the step-by-step (clock cycle by clock cycle) process that each instruction goes through from the start of its FETCH phase to the end of that instruction. Each node in the state machine corresponds to the activity that the processor will carry out during a single clock cycle. The actual processing that is performed in the data path is contained inside the node. The step-by-step flow is conveyed by the arcs that take the processor from each state to the next.

For example, recall that the FETCH phase of every instruction cycle starts with a memory access to read the instruction at the address specified by the PC. Note that in the state numbered 18, the MAR is loaded with the address contained in PC, the PC is incremented by two in preparation for the FETCH of the next LC-3b instruction, and the flow passes to the state numbered 33. The PC is incremented by two since each 16 bit instruction is stored in two consecutive byte-addressable memory locations.

Before we get into what happens during the clock cycle when the processor is in the state numbered 33, we should explain the numbering system – that is, why 18 and 33. Recall, from your knowledge of finite state machines, each state must be uniquely specified and that this unique specification is accomplished by means of the state variables. Our state machine that implements the base LC-3b microarchitecture requires 31 distinct states to describe the entire behavior of the LC-3b base machine. We will come into contact with all of them as we go through this Appendix. Since k logical variables can uniquely identify $2^k$ items, five state variables are sufficient to uniquely specify 31 states. We have chosen six state variables to provide you with enough additional states to handle interrupts, exceptions and virtual memory later in the semester. The number next to each node in Figure C.2 is the decimal equivalent of the values (0 or 1) of the six state variables for the corresponding state. Thus, the state numbered 18 has state variable values 010010.

Now, then, back to what happens after the clock cycle in which the activity of state 18 has finished. Again, if no external device is requesting an interrupt, the flow passes to state 33. In state 33, since the MAR contains the address of the instruction to be processed, this instruction is read from memory and loaded into the MDR. Since this memory access can take multiple cycles, this state continues to execute until a ready signal from the memory (R) is asserted, indicating that the memory access has completed. Thus the MDR contains the valid contents of the memory location specified by MAR. The state machine then moves on to state 35, where the instruction is loaded into the instruction register (IR), completing the fetch phase of the instruction cycle.

Note that the arrow from the last state of each instruction cycle (i.e., the state that completes the processing of that LC-3b instruction) takes us to state 18 (to begin the instruction cycle of the next LC-3b instruction).

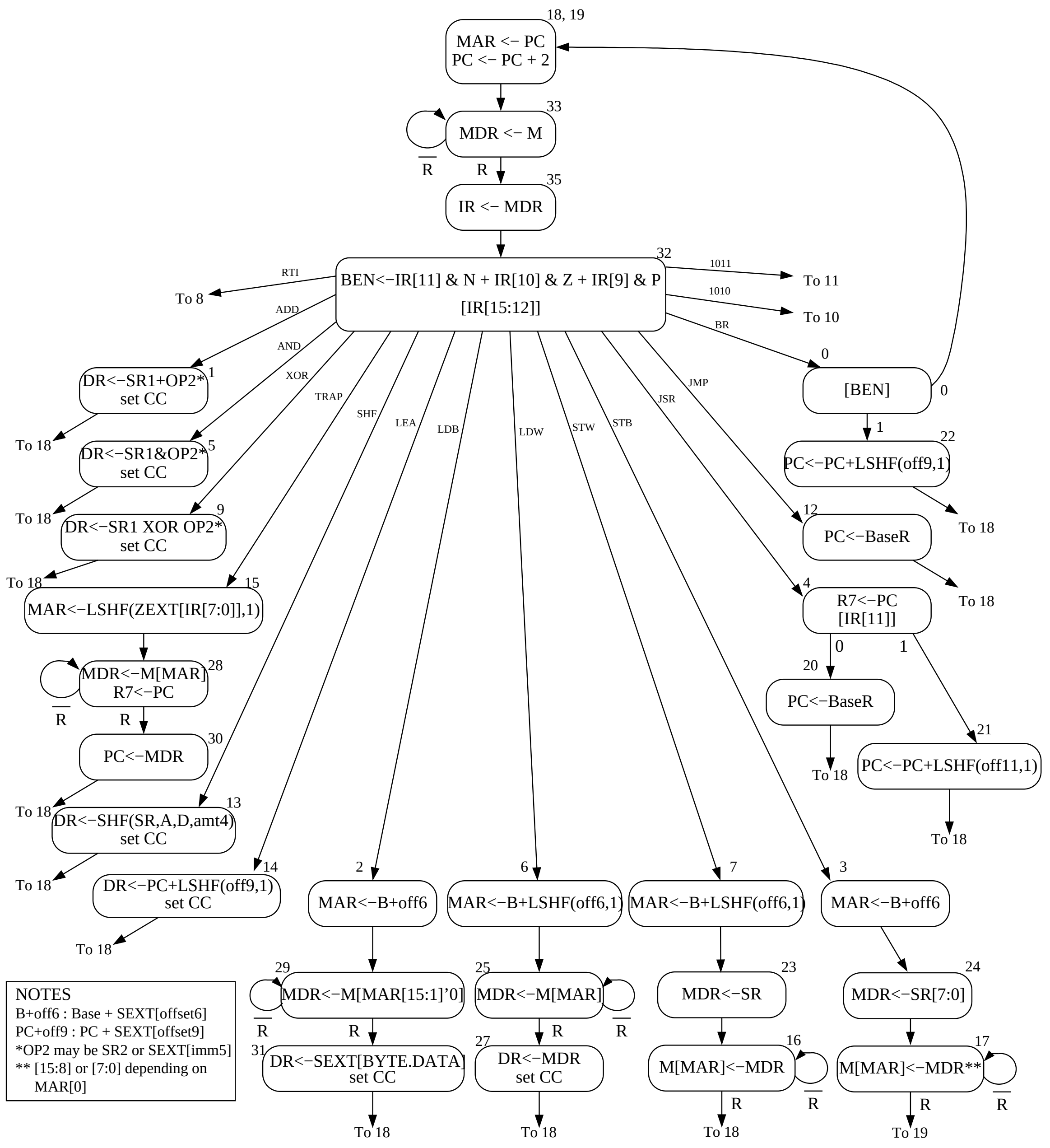

Figure C.2: A state machine for the LC-3b

## C.3 The Data Path

The data path consists of all components that actually process the information during a cycle—the functional units (e.g., the ALU) that operate on the information, the registers that store information at the end of one cycle so it will be available for further use in subsequent cycles, and the buses and wires that carry information from one point to another in the data path. Figure C.3 illustrates the data path of our microarchitecture for the LC-3b.

Note the control signals that are associated with each component in the data path. For example, ALUK, consisting of two control signals, is associated with the ALU. These control signals determine how the component will be used each cycle. Table C.1 lists the set of control signals that control the elements of the data path, and the set of values that each control signal can have. (Actually, for readability, we list a symbolic name for each value, rather than the binary value.) For example, since ALUK consists of two bits, it can have one of four values. Which value it has during any particular clock cycle depends on whether the ALU is required to ADD, AND, XOR, or simply pass one of its inputs to the output during that clock cycle. PCMUX also consists of two control signals and specifies which of the three inputs to the MUX (PC+2, the output of the adder, or whatever has been gated to the bus) is required during a given clock cycle. LD.PC is a single-bit control signal, and is a 0 (NO) or a 1 (YES), depending on whether or not the PC is to be loaded during the given clock cycle.

During each clock cycle, corresponding to the "current state" in the state machine, the 26 bits of control direct the processing of all components in the data path that are required during that clock cycle. The processing that takes place in the data path during that clock cycle, as we have said, is specified inside the node representing that state.

## C.4 The Control Structure

As described above, the state machine determines which control signals are needed to process information in the data path during each clock cycle. The state machine also determines which control signals are needed to direct the flow of control from each state to its successor state.

Figure C.4 shows a block diagram of the control structure of our implementation of the LC-3b. Many implementations are possible, and the design considerations that must be studied to determine which of many possible implementations should be used is the subject of much of this course.

We have chosen here, at the outset, a very straightforward microprogrammed implementation. The current state of the control structure is represented by the 26 bits that control the processing in the data path and the 9 bits that help determine which state comes next. These 35 bits are collectively known as a *microinstruction*. Each microinstruction (i.e., each state of the control structure) is stored in one 35-bit location in a special memory called the control store. Since there are 31 states in the state machine, and since each state corresponds to one microinstruction stored in the control store, the control store for our microprogrammed implementation requires five bits to specify the address of each microinstruction. However, as we have already said, we elected to

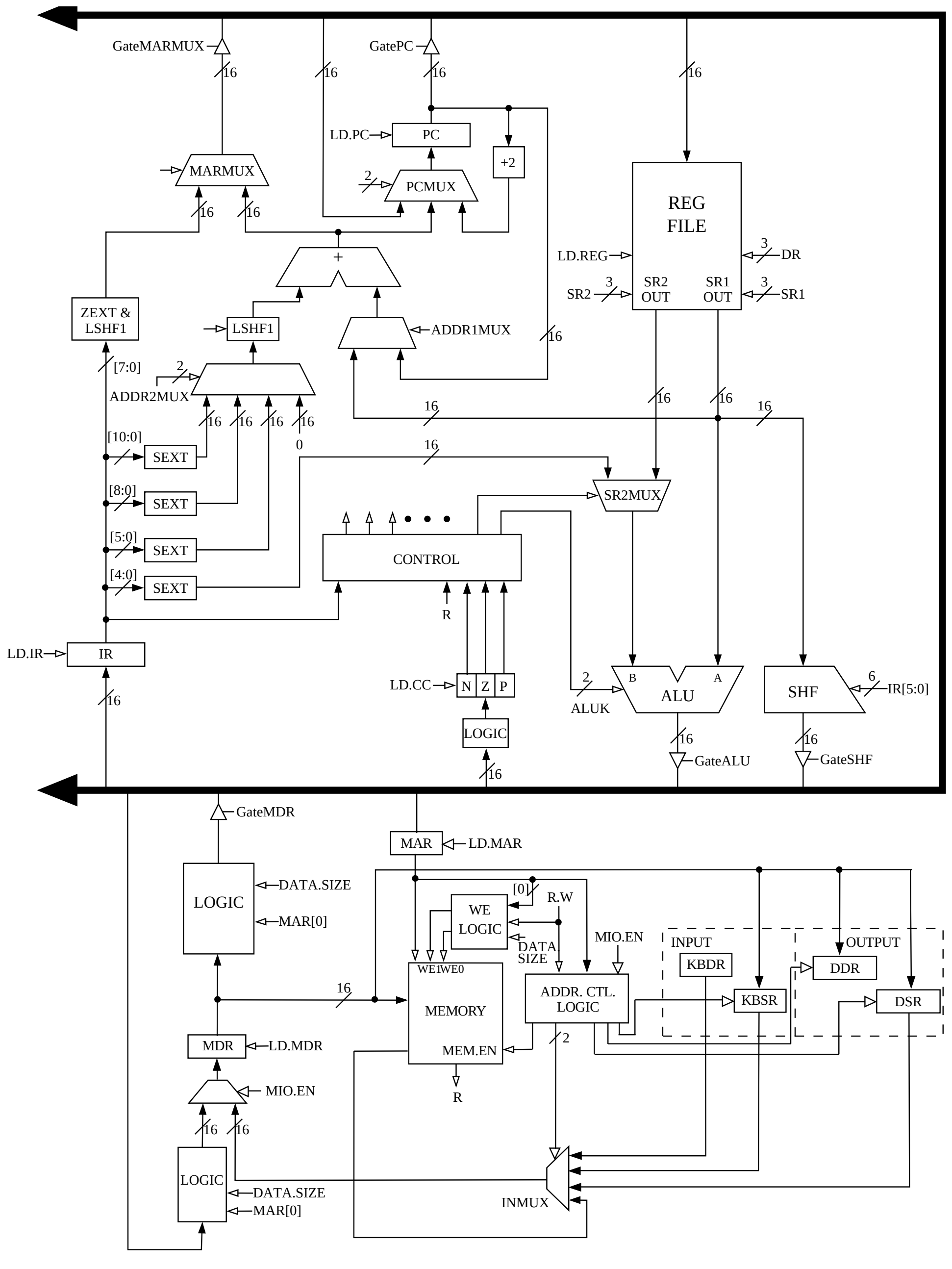

Figure C.3: The LC-3b data path

provide you with the additional flexibility of more states, so we have selected a control store consisting of $2^6$ locations.

| **Signal Name** | **Signal Values** | |
|---|---|---|
| LD.MAR/1: | NO, LOAD | |
| LD.MDR/1: | NO, LOAD | |
| LD.IR/1: | NO, LOAD | |
| LD.BEN/1: | NO, LOAD | |
| LD.REG/1: | NO, LOAD | |
| LD.CC/1: | NO, LOAD | |
| LD.PC/1: | NO, LOAD | |
| GatePC/1: | NO, YES | |
| GateMDR/1: | NO, YES | |
| GateALU/1: | NO, YES | |
| GateMARMUX/1: | NO, YES | |
| GateSHF/1: | NO, YES | |
| PCMUX/2: | PC+2 | ;select pc+2 |
| | BUS | ;select value from bus |
| | ADDER | ;select output of address adder |
| DRMUX/1: | 11.9 | ;destination IR[11:9] |
| | R7 | ;destination R7 |
| SR1MUX/1: | 11.9 | ;source IR[11:9] |
| | 8.6 | ;source IR[8:6] |
| ADDR1MUX/1: | PC, BaseR | |
| ADDR2MUX/2: | ZERO | ;select the value zero |
| | offset6 | ;select SEXT[IR[5:0]] |
| | PCoffset9 | ;select SEXT[IR[8:0]] |
| | PCoffset11 | ;select SEXT[IR[10:0]] |
| MARMUX/1: | 7.0 | ;select LSHF(ZEXT[IR[7:0]],1) |
| | ADDER | ;select output of address adder |
| ALUK/2: | ADD, AND, XOR, PASSA | |
| MIO.EN/1: | NO, YES | |
| R.W/1: | RD, WR | |
| DATA.SIZE/1: | BYTE, WORD | |
| LSHF1/1: | NO, YES | |

Table C.1: Data path control signals

Table C.2 lists the function of the 9 bits of control information that help determine which state comes next. Figure C.5 shows the logic of the microsequencer. The purpose of the microsequencer is to determine the address in the control store that corresponds to the next state, that is, the location where the 35 bits of control information for the next state are stored.

Note that state 32 of the state machine (Figure C.2) has 16 "next" states, depending

| **Signal Name** | **Signal Values** | |
|---|---|---|
| J/6: | | |
| COND/2: | $COND_0$ | ;Unconditional |
| | $COND_1$ | ;Memory Ready |
| | $COND_2$ | ;Branch |
| | $COND_3$ | ;Addressing Mode |
| IRD/1: | NO, YES | |

Table C.2: Microsequencer control signals

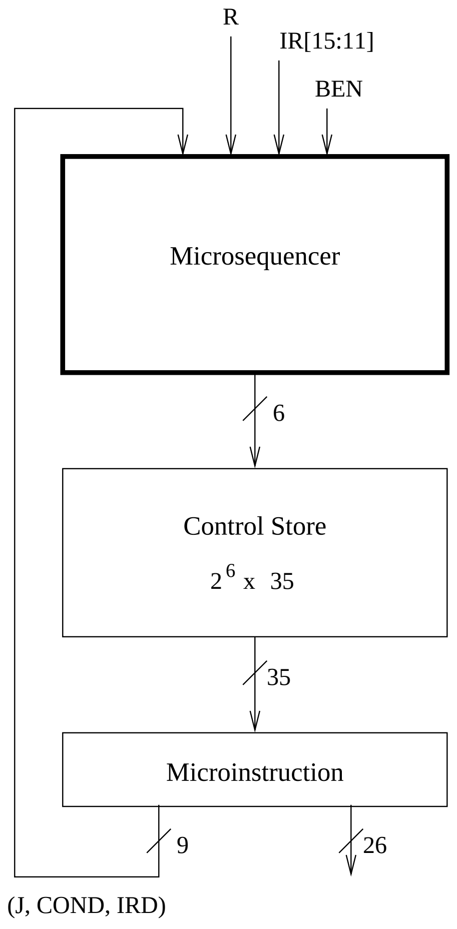

Figure C.4: The control structure of a microprogrammed implementation, overall block diagram

on the LC-3b instruction being executed during the current instruction cycle. This state carries out the DECODE phase of the instruction cycle. If the IRD control signal in the microinstruction corresponding to state 32 is 1, the output MUX of the microsequencer (Figure C.5) will take its source from the six bits formed by 00 concatenated with the four opcode bits IR[15:12]. Since IR[15:12] specifies the opcode of the current LC-3b instruction being processed, the next address of the control store will be one of 16 addresses, corresponding to the 14 opcodes plus the two unused opcodes, IR[15:12] = 1010 and 1011. That is, each of the 16 next states is the first state to be carried out after the instruction has been decoded in state 32. For example, if the instruction being processed is ADD, the address of the next state is state 1, whose microinstruction is stored at location 000001. Recall that IR[15:12] for ADD is 0001.

If, somehow, the instruction inadvertently contained IR[15:12] = 1010 or 1011, the

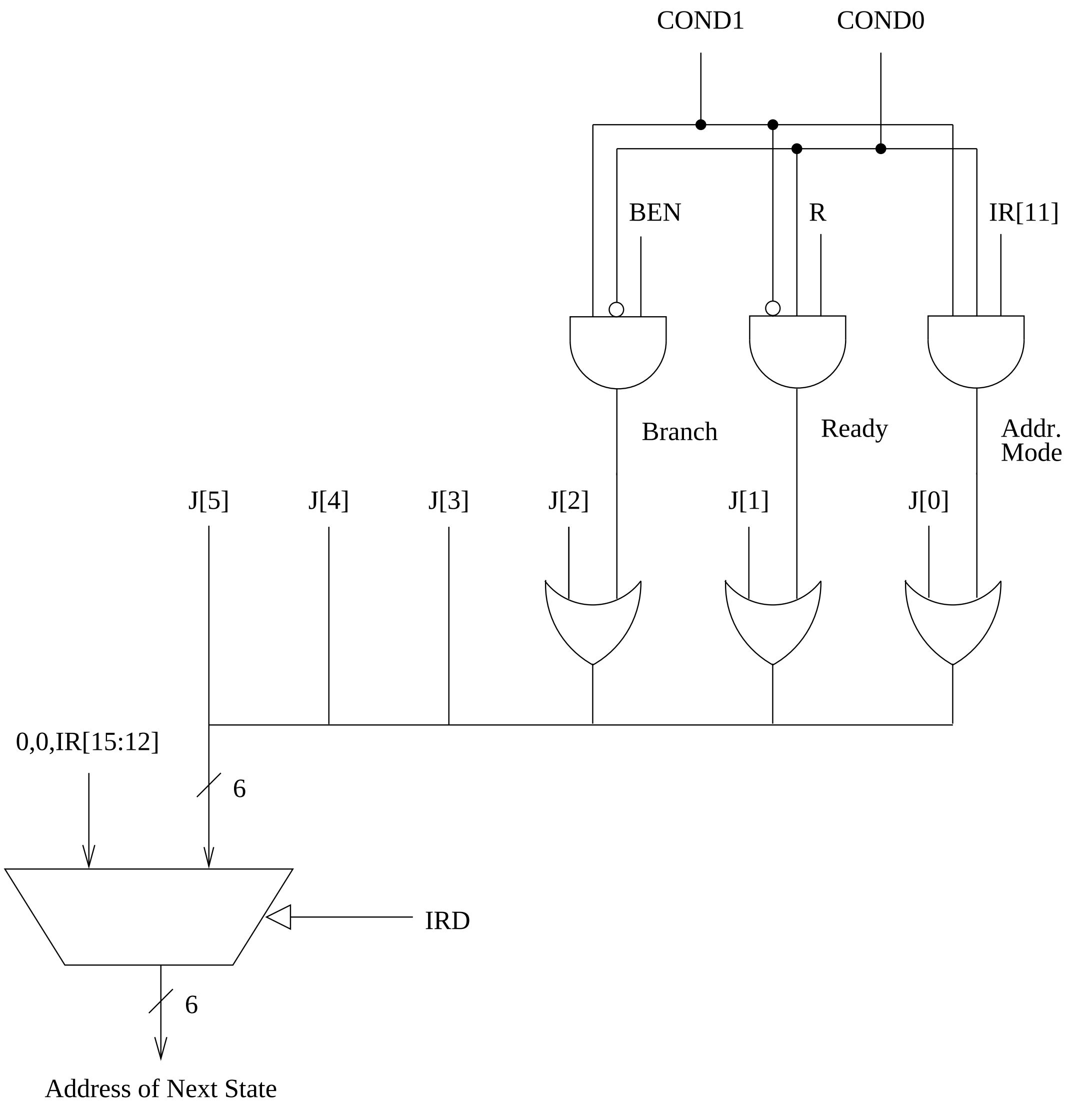

Figure C.5: The microsequencer of the LC-3b base machine

unused opcodes, the microarchitecture would execute a sequence of microinstructions, starting at state 10 or state 11, depending on which illegal opcode was being decoded. In both cases, the sequence of microinstructions would respond to the fact that an instruction with an illegal opcode had been fetched.

Several signals necessary to control the data path and the microsequencer are not among those listed in Tables C.1 and C.2. They are DR, SR1, BEN, and R. Figure C.6 shows the additional logic needed to generate DR, SR1, and BEN.

The remaining signal, R, is a signal generated by the memory in order to allow the

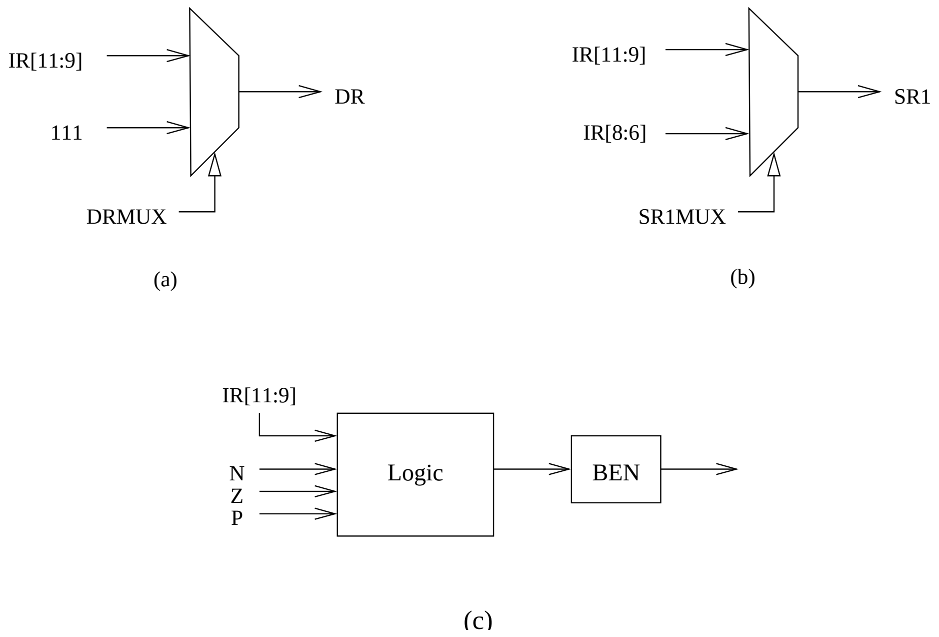

Figure C.6: Additional logic required to provide control signals

LC-3b to operate correctly with a memory that takes multiple clock cycles to read or store a value.

Suppose it takes memory five cycles to read a value. That is, once MAR contains the address to be read and the microinstruction asserts READ, it will take five cycles before the contents of the specified location in memory are available to be loaded into MDR. (Note that the microinstruction asserts READ by means of three control signals: MIO.EN/YES, R.W/RD, and DATA.SIZE/WORD; see Figure C.3.)

Recall our discussion in Section C.2 of the function of state 33, which accesses an instruction from memory during the fetch phase of each instruction cycle. For the LC-3b to operate correctly, state 33 must execute five times before moving on to state 35. That is, until MDR contains valid data from the memory location specified by the contents of MAR, we want state 33 to continue to re-execute. After five clock cycles, the memory has completed the "read," resulting in valid data in MDR, so the processor can move on to state 35. What if the microarchitecture did not wait for the memory to complete the read operation before moving on to state 35? Since the contents of MDR would still be garbage, the microarchitecture would put garbage into IR in state 35.

The ready signal (R) enables the memory read to execute correctly. Since the memory knows it needs five clock cycles to complete the read, it asserts a ready signal (R) throughout the fifth clock cycle. Figure C.2 shows that the next state is 33 (i.e., 100001) if the memory read will not complete in the current clock cycle and state 35 (i.e., 100011) if it will. As we have seen, it is the job of the microsequencer (Figure C.5) to produce the next state address.

The 9 microsequencer control bits for state 33 are as follows:

```
IRD/0       ; NO
COND/01     ; Memory Ready
J/100001
```

With these control signals, what next state address is generated by the microsequencer? For each of the first four executions of state 33, since R $= 0$, the next state address is 100001. This causes state 33 to be executed again in the next clock cycle. In the fifth clock cycle, since R $= 1$, the next state address is 100011, and the LC-3b moves on to state 35. Note that in order for the ready signal (R) from memory to be part of the next state address, COND had to be set to 01, which allowed R to pass through its three-input AND gate.

## C.5 Alignment correction for Byte Loads and Stores

Everything in the discussion thus far has involved word accesses from memory. Because the LC-3b is byte-addressable, and loads and stores can access either byte or word data, additional support is required from both the data path and the microsequencer. The only memory read that is accessing a byte of data is state 29 in the state machine. The only memory store that is writing a byte of data is state 17 in the state machine. Support is provided for both in the data path as follows.

### C.5.1 Byte loads in state 29

In state 29, 16 bits are read from memory as usual, and loaded into MDR. In state 31, the data read is loaded into the destination register as specified by bits[11:9] of the LDB instruction, as follows: A MUX selects whether MDR[15:8] or MDR[7:0] is the correct byte to be loaded, based on the low bit of the address (MAR[0]). This byte is sign-extended to 16 bits. A second MUX selects either this sign-extended byte of data or the word in MDR, based on the control signal DATA.SIZE. Since the instruction being processed is LDB, state 31 has the control signal DATA.SIZE/BYTE. The output of this MUX (the sign-extended byte of data) is gated onto the bus and loaded into DR.

### C.5.2 Byte stores in state 17

In state 24, just prior to state 17 which does the actual byte store, the data to be stored is loaded into MDR as follows: If MAR[0]=1, SR[7:0] must be loaded into the odd address specified by MAR. A MUX selects either SR[15:0] or SR[7:0]'SR[7:0], based on MAR[0]. In that way, if the instruction being processed is STW, MAR[0] must be 0, and the store proceeds fine. If the instruction being processed is STB, SR[7:0] is in MDR[7:0] if MAR[0]=0, and in MDR[15:8] if MAR[0]=1. That is, the data in MDR is properly aligned ready to be stored.

In state 17, the actual store takes place as follows: Two write enable signals WE1 and WE0 control the stores to the odd and even addresses of a memory word. WE1 controls bits [15:8] and WE0 controls bits [7:0] of the same word of memory. Which write enable signals are asserted depends on R.W, DATA.SIZE, and MAR[0]. Write enable signals are only asserted if the machine is doing a store. Ergo, R.W must be WR. If DATA.SIZE is BYTE, MAR[0] determines whether WE1 or WE0 is asserted. Recall that if DATA.SIZE is BYTE, MDR was previously loaded with SR[7:0]'SR[7:0]. If MAR[0]=0, WE0 is asserted and MDR[7:0] (i.e., SR[7:0]) is written to memory. If MAR[0]=1, WE1 is asserted and MDR[15:8] (i..e, SR[7:0]) is written to memory. Thus in both cases, the relevant byte is stored to the correct location in memory.

If DATA.SIZE is WORD and MAR[0]=0, then WE1 and WE0 are both asserted and the word in MDR is written to memory. If DATA.SIZE is WORD and MAR[0]=1, an illegal operand address exception would have been taken earlier in the microsequence.

Once the write completes, Memory Ready is asserted and control passes from state 17 to state 19. State 19 is an exact duplicate of state 18. State 18 and 19 then begin the processing of the next LC-3b instruction.

## C.6 Memory-mapped I/O

As you know from Chapter 8, the LC-3b ISA performs input and output via memory-mapped I/O, that is, with the same data movement instructions that it uses to read from and write to memory. The LC-3b does this by assigning an address to each device register. Input is accomplished by a load instruction whose effective address is the address of an input device register. Output is accomplished by a store instruction whose effective address is the address of an output device register. For example, in state 25 of Figure C.2, if the address in MAR is xFE02, MDR is supplied by the KBDR, and the data input will be the last keyboard character typed. On the other hand, if the address in MAR is a legitimate memory address, MDR is supplied by the memory.

The state machine of Figure C.2 does not have to be altered to accommodate memory-mapped I/O. However, something has to determine when memory should be accessed and when I/O device registers should be accessed. This is the job of the address control logic shown in Figure C.3.

The control signals that are generated, are based on (1) the contents of MAR, (2) whether or not memory or I/O is accessed this cycle (MIO.EN/NO, YES), and (3) whether a load or store is requested (R.W/Read, Write). One of your tasks in problem set 2 will be to generate the truth table for this block. Incidientally, the device registers are all 16 bit registers, and have even addresses. They are accessed by LDW and STW instructions. This eliminates all alignment problems on I/O accesses.

## C.7 Control Store

Figure C.7 completes our microprogrammed implementation of the LC-3b. It shows the contents of each location of the control store, corresponding to the 35 control signals required by each state of the state machine. We have left the exact entries blank to allow you, dear reader, the joy of filling in the required signals yourself.

| IRD | Cond | J | LD.MAR | LD.MDR | LD.IR | LD.BEN | LD.REG | LD.CC | LD.PC | GatePC | GateMDR | GateALU | GateMARMUX | GateSHF | PCMUX | DRMUX | SR1MUX | ADDR1MUX | ADDR2MUX | MARMUX | ALUK | MIO.EN | R.W | DATA.SIZE | LSHF1 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | | | | | | | | | | | 000000 (State 0) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 000001 (State 1) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 000010 (State 2) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 000011 (State 3) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 000100 (State 4) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 000101 (State 5) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 000110 (State 6) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 000111 (State 7) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 001000 (State 8) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 001001 (State 9) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 001010 (State 10) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 001011 (State 11) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 001100 (State 12) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 001101 (State 13) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 001110 (State 14) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 001111 (State 15) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 010000 (State 16) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 010001 (State 17) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 010010 (State 18) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 010011 (State 19) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 010100 (State 20) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 010101 (State 21) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 010110 (State 22) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 010111 (State 23) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 011000 (State 24) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 011001 (State 25) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 011010 (State 26) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 011011 (State 27) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 011100 (State 28) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 011101 (State 29) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 011110 (State 30) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 011111 (State 31) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 100000 (State 32) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 100001 (State 33) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 100010 (State 34) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 100011 (State 35) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 100100 (State 36) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 100101 (State 37) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 100110 (State 38) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 100111 (State 39) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 101000 (State 40) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 101001 (State 41) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 101010 (State 42) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 101011 (State 43) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 101100 (State 44) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 101101 (State 45) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 101110 (State 46) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 101111 (State 47) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 110000 (State 48) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 110001 (State 49) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 110010 (State 50) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 110011 (State 51) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 110100 (State 52) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 110101 (State 53) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 110110 (State 54) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 110111 (State 55) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 111000 (State 56) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 111001 (State 57) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 111010 (State 58) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 111011 (State 59) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 111100 (State 60) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 111101 (State 61) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 111110 (State 62) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | 111111 (State 63) |

Figure C.7: Specification of the control store