The LC3 Datapath (Chapter 5, Appendix B,C)

1

The LC-3 ISA: summary

- 16 bit instructions and data
  - 2's complement data type
- Operate/ALU instructions: ADD, NOT, AND
- Data movement Inst: Load and Store
  - Addressing mode: PC-relative, Indirect, Register/Base+Offset
- Transfer of control instructions
  - Branch – using condition code registers
  - Jump – unconditional branch
  - Traps, Subroutine calls – discuss later
- LC3 Assembly Language and assembler
- Let' take a look at the LC3 datapath and control unit design

2

2

LC3 Datapath – from Logic to Processor Data Path

•The data path of a computer is all the logic used to process information.

• Take all the devices we have discussed and use them to build a circuit that implements a von Neumann machine

•Combinational Logic

- Decoders -- convert instructions into control signals
- Multiplexers -- select inputs and outputs
- ALU (Arithmetic and Logic Unit) -- operations on data

•Sequential Logic

- State machine -- coordinate control signals and data movement
- Registers and latches -- storage elements

3

3

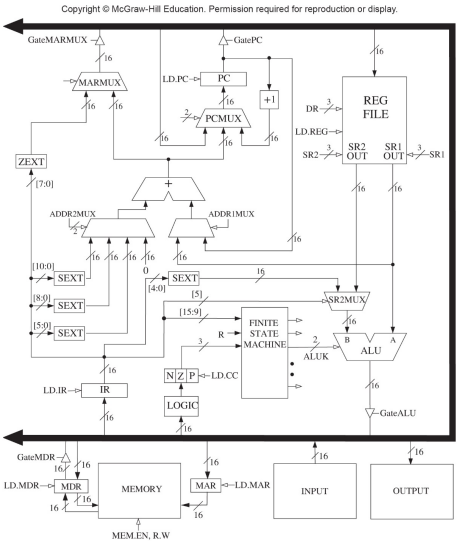
Implementation on LC3 datapath

- Components/paths of the LC3 datapath need to be activated to implement an instruction
  - our dataflow diagrams describe the devices and paths used to implement an instruction
- Key components:
  - Global Bus – tristate devices to control access to bus
  - Memory – MAR, MDR
  - ALU – outputs to bus, inputs from register/IR
  - Register file – select Source registers, Dest registers, input bus
  - PC + PCMUX: determine address of next instruction
  - MAR + MARMUX: where is memory address
  - Condition code logic: 1 bit registers
  - SEXT: sign extension logic
  - Control/Finite state machine?
  - Multiplexers: SR2MUX, ADDR1MUX, ADDR2MUX, PCMUX,…

4

4

LC-3 Data Path – The complete datapath



Data path is used to execute LC-3 programs.

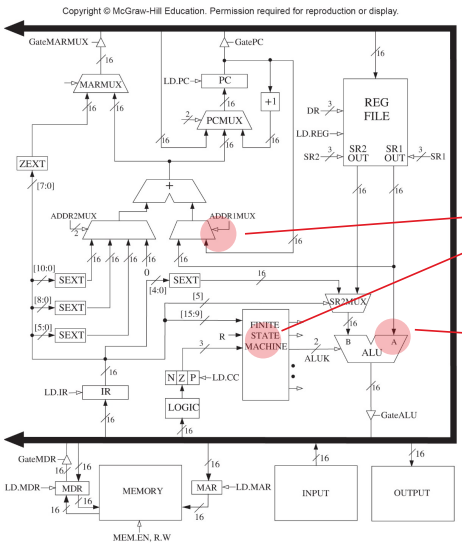
PC is initialized to point to the first instruction. Clock is enabled, and the control unit takes over.

Next slides will give a little more detail on various components.

5

5

LC-3 Data Path - Components



Arrows with open heads represent control signals from FSM –the control unit!

Arrows with filled heads represent data that is processed.

6

6

A Useful Analogy

- The datapath corresponds to the *tracks* in a railway
  - pathways that allow you to move information around the CPU
- The control signals control the *switches* that connect tracks
  - Signals that setup the pathways so data can flow through CPU

7

7

Data Path Components

•Global bus

- special set of wires that carry a 16-bit signal to many components
- inputs to the bus are "tri-state devices," that only place a signal on the bus when they are enabled
- only one (16-bit) signal should be enabled at any time
  - control unit decides which signal "drives" the bus
- any number of components can read the bus
  - register only captures bus data if it is write-enabled by the control unit

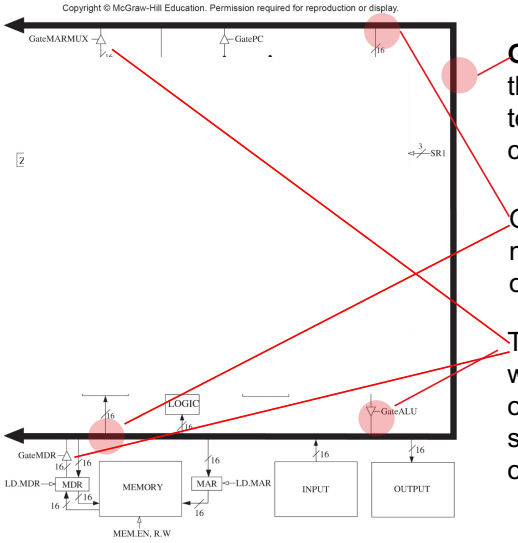
•Memory

- Control and data registers for memory and I/O devices
- memory: MAR, MDR (also control signal for read/write)

8

8

LC-3 Data Path- Components



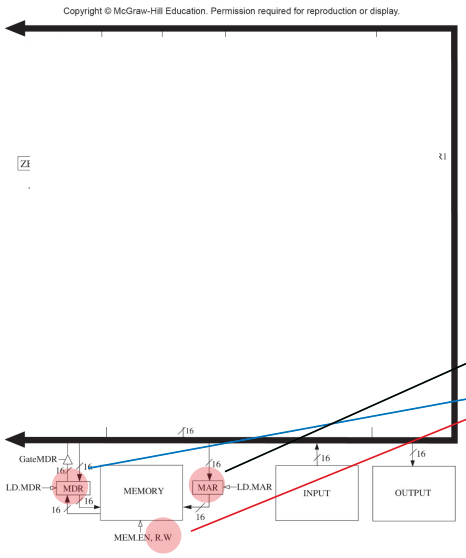
Global bus is a set of wires that allow various components to transfer 16-bit data to other components.

One or more components may read data from the bus on any cycle.

Tri-state device determines which component puts data on the bus. Only one source of data at any time.

9

LC-3 Data Path



Memory interface:
MAR
MDR
Read/Write control

10

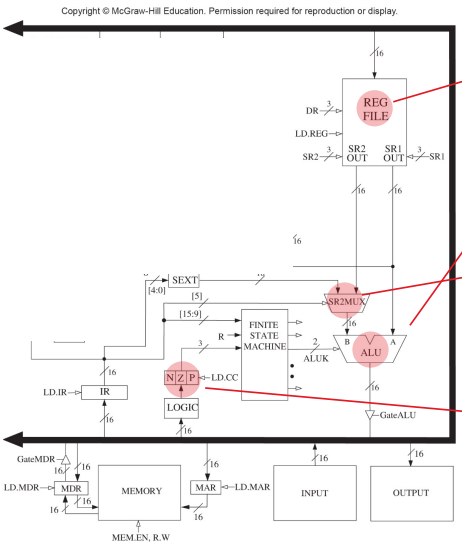
Data Path Components

- ALU
  - Accepts inputs from register file and from sign-extended bits from IR (immediate field).
    - Bit 5 of LC3 instruction determines this
  - Output goes to bus.
    - used by condition code logic, register file, memory
  - Function to apply: determined by opcode – need 2 bits ALUK
- Register File
  - Two read addresses (SR1, SR2), one write address (DR)
  - Input from bus
    - result of ALU operation or memory read
  - Two 16-bit outputs
    - used by ALU, PC, memory address
    - data for store instructions passes through ALU

11

11

LC-3 Data Path



Register File (R0-R7) Control signals specify two source register (SR1, SR2) and one destination (DR).

ALU performs ADD, AND, NOT. Operand A always comes from register file. Operand B is from register file or IR. Output goes to bus, to be written into register file.

Condition codes are set by looking at data placed on the bus by ALU or memory (MDR).

12

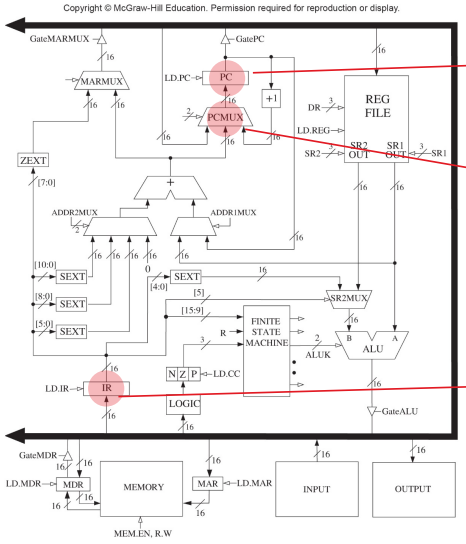
Data Path Components

- PC and PCMUX
  - Three inputs to PC, controlled by PCMUX
    1. PC+1 – FETCH stage
    2. Address adder – BR, JMP
    3. bus – TRAP (discussed later)

- MAR and MARMUX
  - Two inputs to MAR, controlled by MARMUX
    1. Address adder – LD/ST, LDR/STR
    2. Zero-extended IR[7:0] -- TRAP (discussed later)

13

LC-3 Data Path



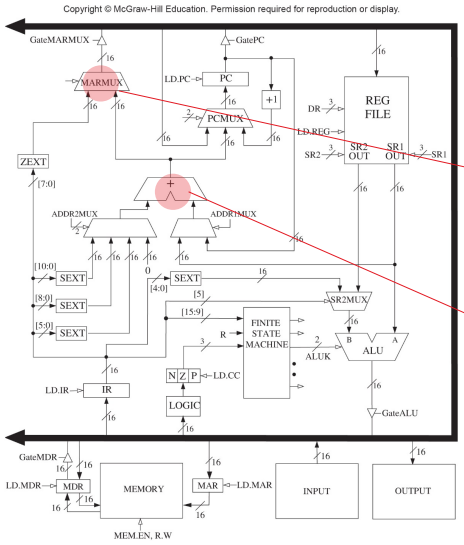
PC puts address on bus. Placed in MAR during Fetch.

PCMUX allows various values to be written to PC: incremented PC (Fetch), computed address (BR), or register data from bus (JMP).

IR gets data from bus (MDR) during Fetch.

14

LC-3 Data Path



MARMUX chooses value to be written to MAR during load, store, or TRAP.

Evaluate Address phase adds offset to PC or register for load, store, BR.

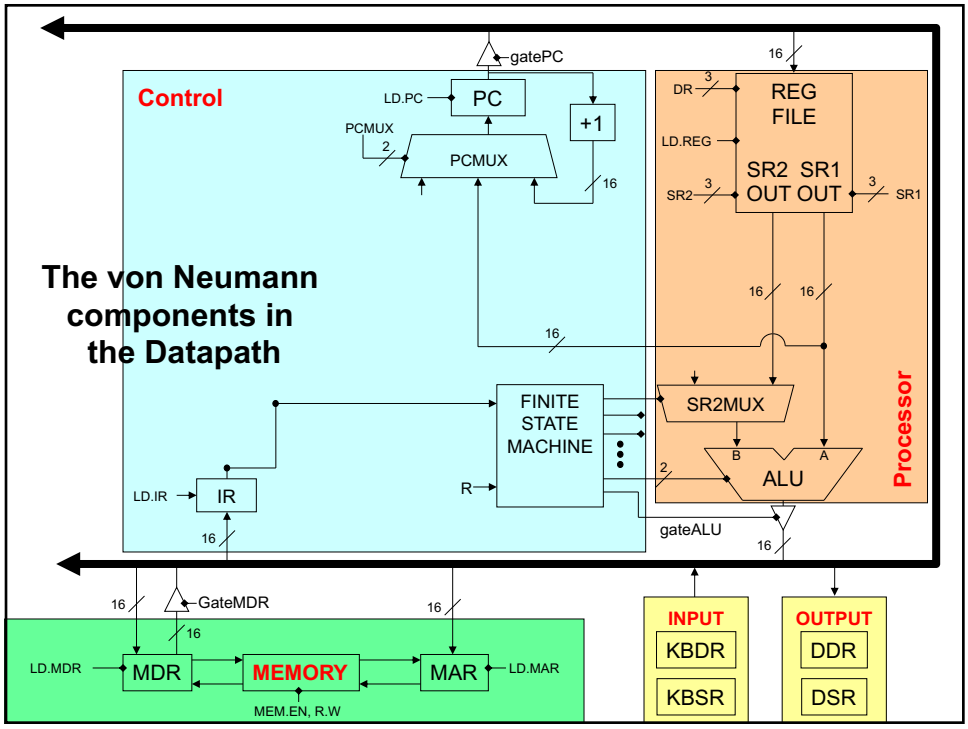
15

Data Path Components

- Condition Code Logic
  - Looks at value on bus and generates N, Z, P signals
  - Registers set only when control unit enables them (LD.CC)
    - only certain instructions set the codes (ADD, AND, NOT, LD, LDI, LDR, LEA)
- Control Unit – Finite State Machine
  - On each machine cycle, changes control signals for next phase of instruction processing
    - who drives the bus? (GatePC, GateALU, …)
    - which registers are write enabled? (LD.IR, LD.REG, …)
    - which operation should ALU perform? (ALUK)
    - …
  - Logic includes decoder for opcode, etc.

16

16



17

Instruction Execution and Datapath & Control Signals

- Examine the instruction execution process and the resulting dataflow
  - What devices are used
  - What control signals are needed to execute the instruction
    - These signals are generated by the Control Unit
      - Implemented (conceptually) as a Finite State Machine
- Recall: instruction execution in LC3 (and all von Neumann) goes through the 6 phase instruction processing cycle

18

18

Instruction Processing Cycle – implementation of LC3 Datapath

Six phases of the complete Instruction Cycle

- 1. Fetch: load IR with instruction from memory
- 2. Decode: determine action to take (set up inputs for ALU, RAM, etc.)
- 3. Evaluate address: compute memory addr of operands, if any
- 4. Fetch operands: read operands from memory or registers
- 5. Execute: carry out instruction
- 6. Store results: write result to destination (register or memory)

19

19

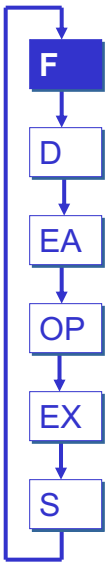
Instruction Processing Step 1: FETCH

•Load next instruction (at address stored in PC) from memory into Instruction Register (IR).

- 1.Copy contents of PC into MAR: MAR ← (PC)
- 2.Send "read" signal to mem and read: MDR ← (MAR)
- 3.Copy contents of MDR into IR: IR ← MDR
- 4. increment PC, so that it points to next inst

  in sequence: PC = PC+1

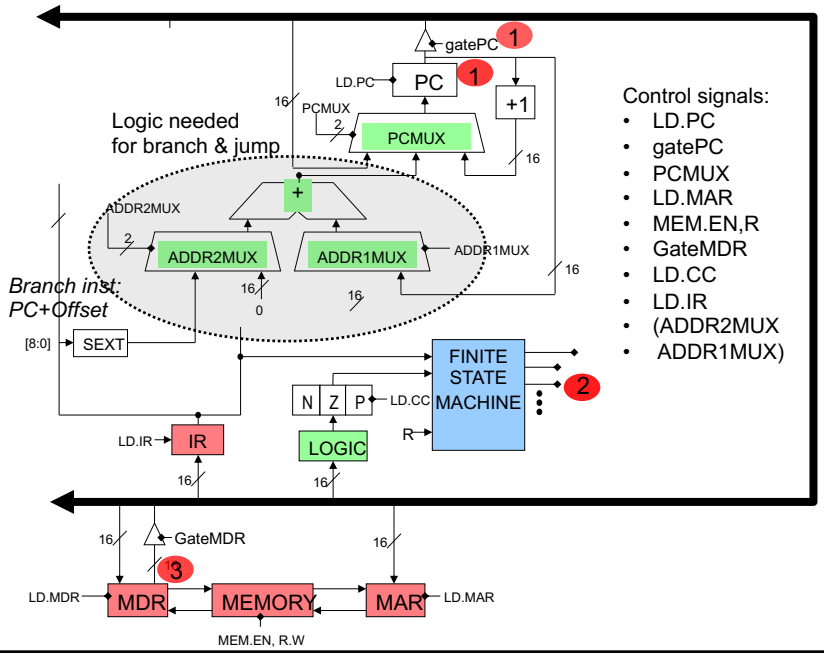
  Step 4 and Step 1 can be done at the same cycle



20

20

LC3: Instruction Fetch



21

21

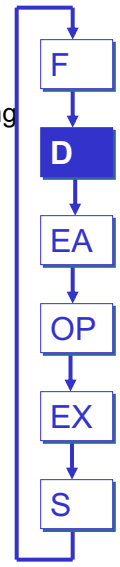
Instruction Processing Step 2: DECODE

•First identify the opcode.

- In LC-3, this is always the first four bits of instruction.
  - A **4-to-16 decoder** asserts a control line corresponding to the desired opcode.

•Depending on opcode, identify other operands from the remaining bits.

- Example:
  - for LDR, last six bits is offset
  - for ADD, last three bits is source operand #2



22

22

Instruction Processing Cycle- Remaining steps

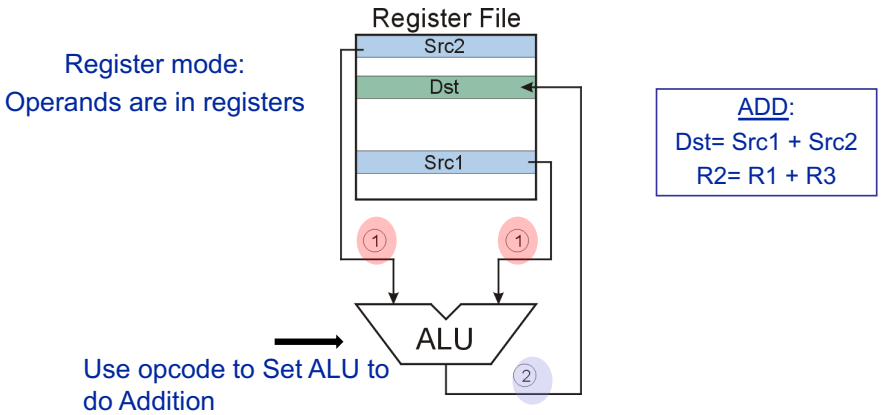
- After Instruction fetch and decode, the next cycles are evaluate address, operand fetch, execute, store.
- To design datapath and control unit:
  - What are the logic devices needed to execute instructions
  - What control signals should be generated by the control unit

- Consider three examples:
  1. ADD instruction execution
  2. LD (evaluate address, operand fetch and store into register)
  3. LDR (evaluate address, operand fetch and store into register)

23

23

ADD



Register mode:
Operands are in registers

ADD:
Dst= Src1 + Src2
R2= R1 + R3

Use opcode to Set ALU to do Addition

24

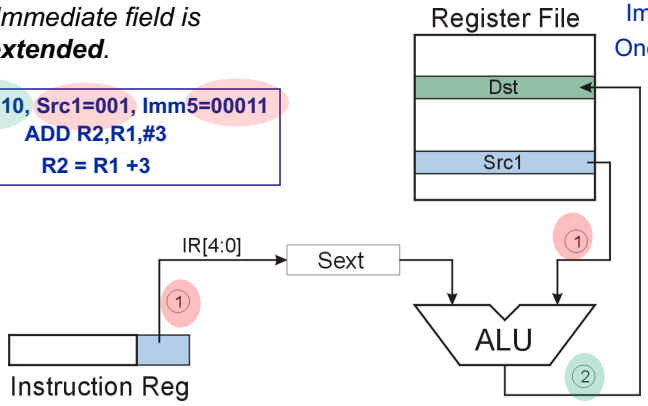
ADD/AND (Immediate)

this one means "immediate mode"

Note: Immediate field is ***sign-extended***.

If Dst=010, Src1=001, Imm5=00011
ADD R2,R1,#3
R2 = R1 +3

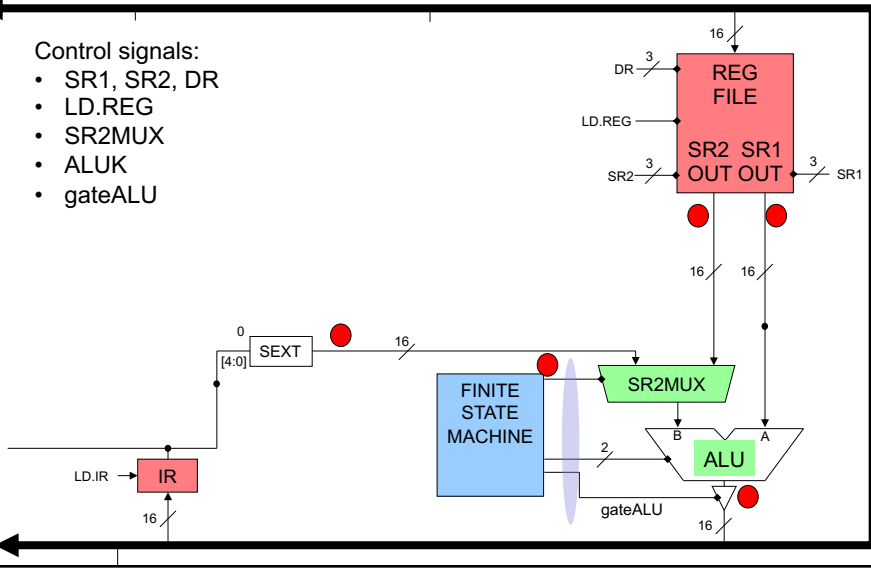
Immediate mode:
One Operand in inst



25

ADD (AND) Instruction: LC3 Datapath

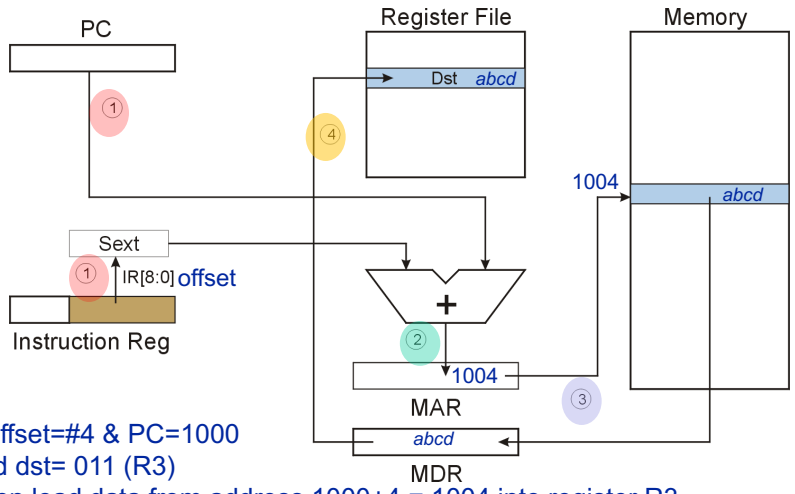
Control signals:
- SR1, SR2, DR
- LD.REG
- SR2MUX
- ALUK
- gateALU



26

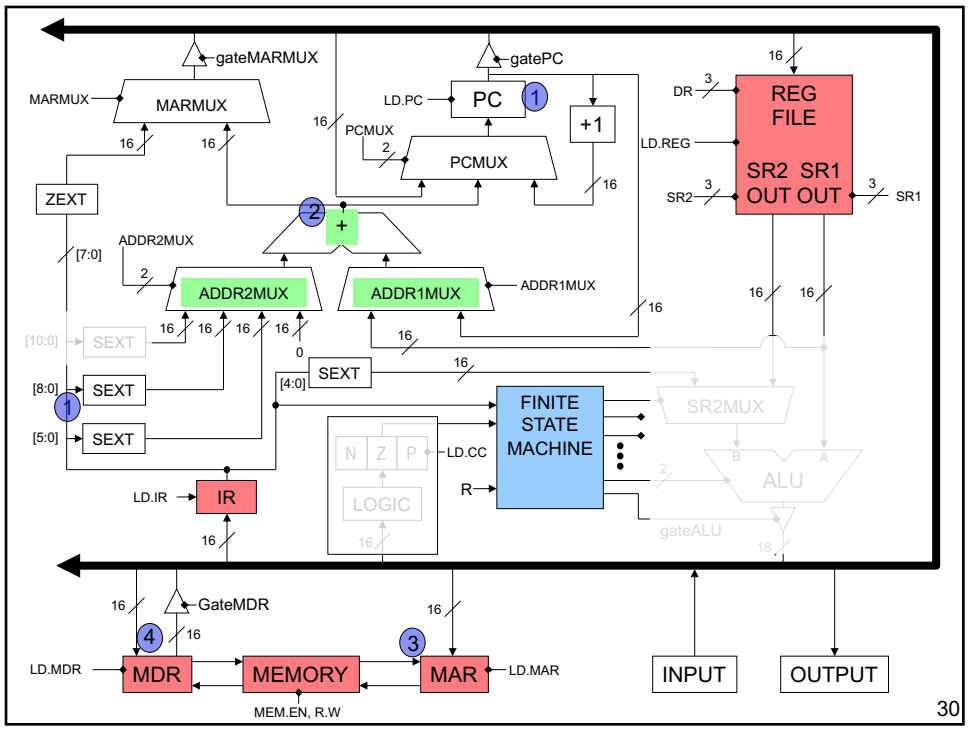
26

LD (PC-Relative)



If offset=#4 & PC=1000
and dst= 011 (R3)
Then load data from address 1000+4 = 1004 into register R3

29

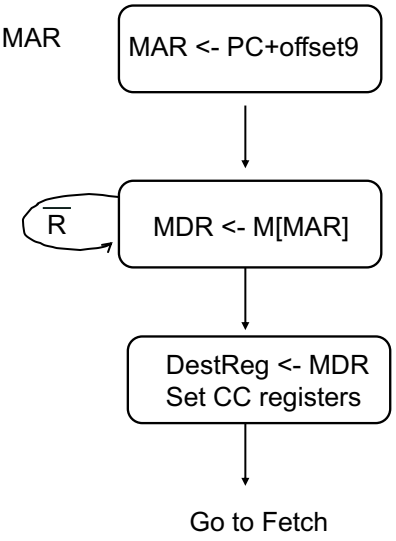


30

30

LD Instruction steps and control signals

- Signals to add PC+offset & load into MAR
  - ADDR1MUX
  - ADDR2MUX
  - MARMUX
  - gateMARMUX
  - LD.MAR
- Signals to Read
  - MEM.EN, R
  - LD.MDR
- Signals to store into DestReg & CC
  - GateMDR
  - DR bits
  - LD.REG
  - LD.CC



31

31

Finite State Machine for Control Unit ?

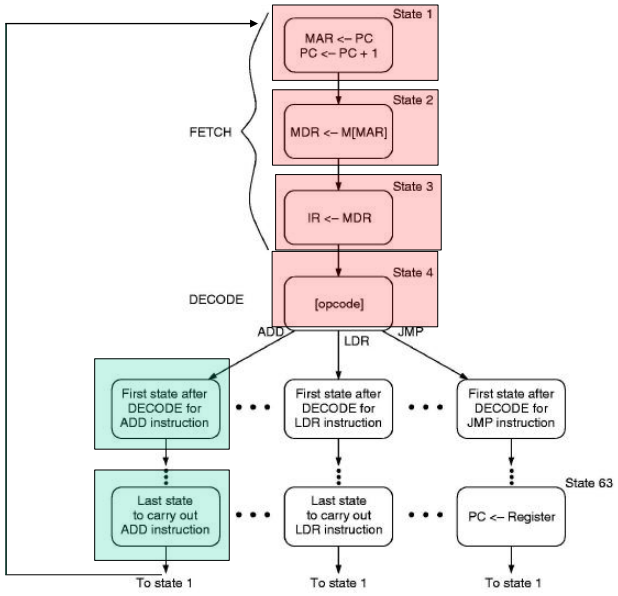
- The process of the instruction execution cycle can be modeled as a finite state machine
- The control unit is a state machine
  - Transition from state to state based on the steps in the instruction cycle, the opcode, and outcome (for branches)
    - Determine the signals to be generated at each phase of the instruction cycle – these are the outputs to be generated by the FSM
  - Appendix C has complete state diagram

32

32

The Instruction Cycle as FSM

A simplified state diagram

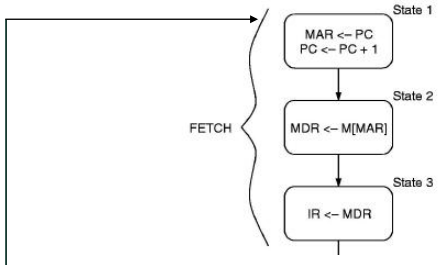


33

33

The Instruction Cycle as FSM

A simplified state diagram



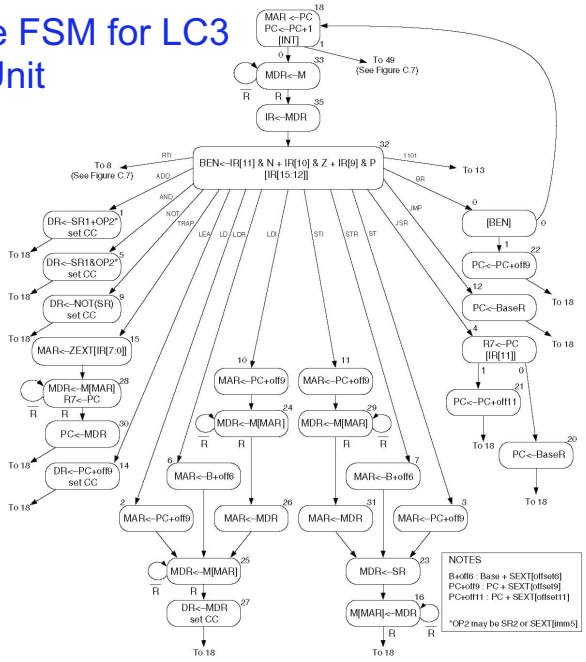
At each state generate the required control signals:

- Refer to datapath to determine which signals
  - 1. gatePC, LD.MAR, LD.CC
  - 2. MEM.EN.R
  - 3. GateMDR, LD.IR

34

34

Complete FSM for LC3 Control Unit



35

35

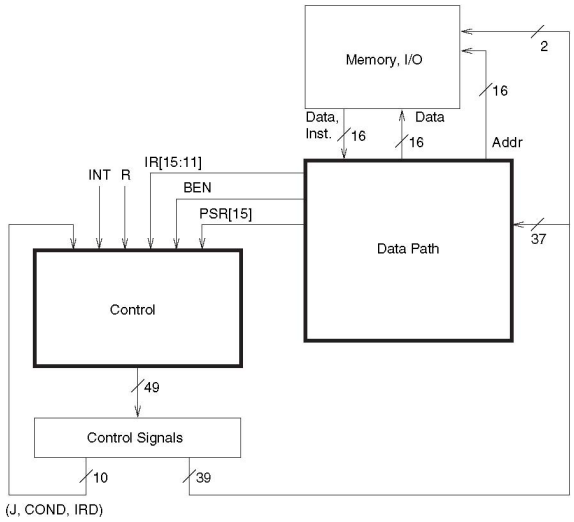
Implementing the Control Logic

- Given the state diagram one can implement the controller in many ways
  - 52 states
  - Each needs 39 control lines plus 10 to determine next state = 49 control lines
- What should controller do ?
  - Generate the 49 control signals at each cycle
- Implement this as a Microprogram
  - Use 6 bit address to get the microinstruction
  - Start state and progress through states based on microinstruction

36

36

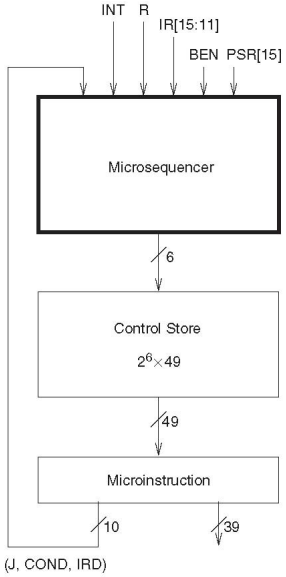
Control Unit, Data Path, and Memory - Interactions



37

37

Microprogrammed Implementation



38

38

Datapath Summary

- Given an instruction set, we saw how each instruction's execution is carried out
  - Requires setting control signals to "route the data"
- The control unit can be implemented as a FSM
  - At each state, it generates the value for the control signals
  - Transitions from state to state in one cycle
  - For simpler implementation, the generation of the control signal by the FSM can be implemented using "microinstructions"
- Designing a processor from scratch ?
  - Example of a simple processor design

39

39

EXAMPLE: BASIC CPU DATAPATH & CONTROL

40