Mc Graw Hill Education

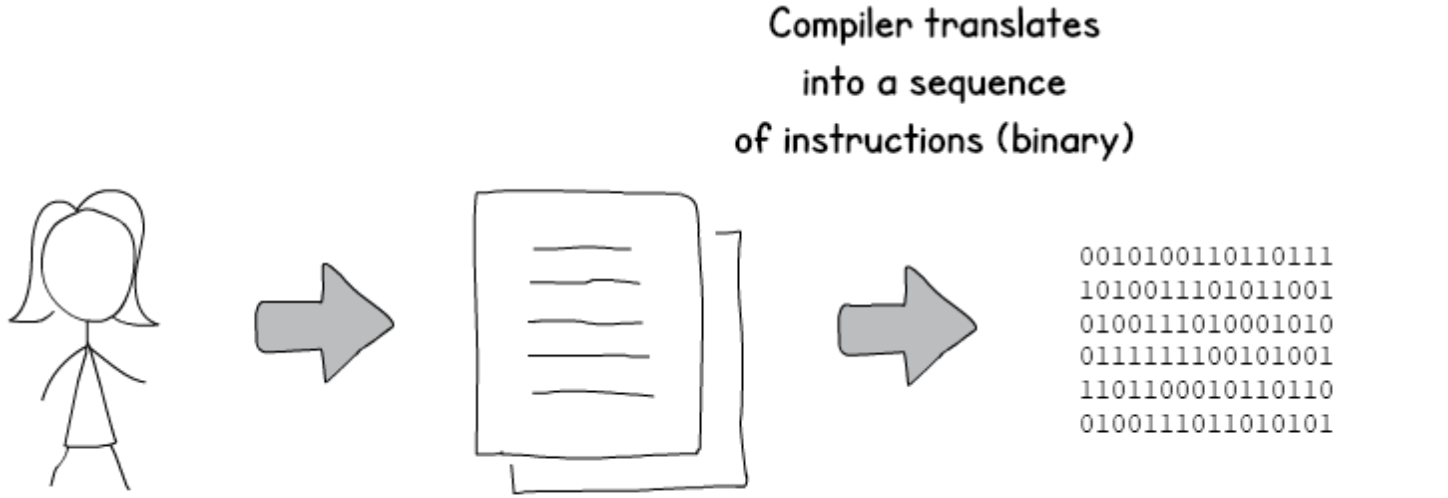
Because learning changes everything.®

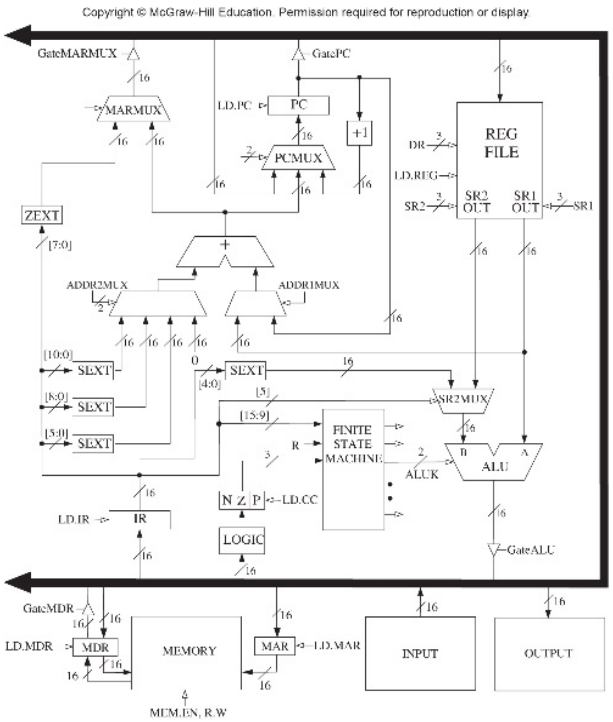
From Bits and Gates to C and Beyond

The von Neumann Model

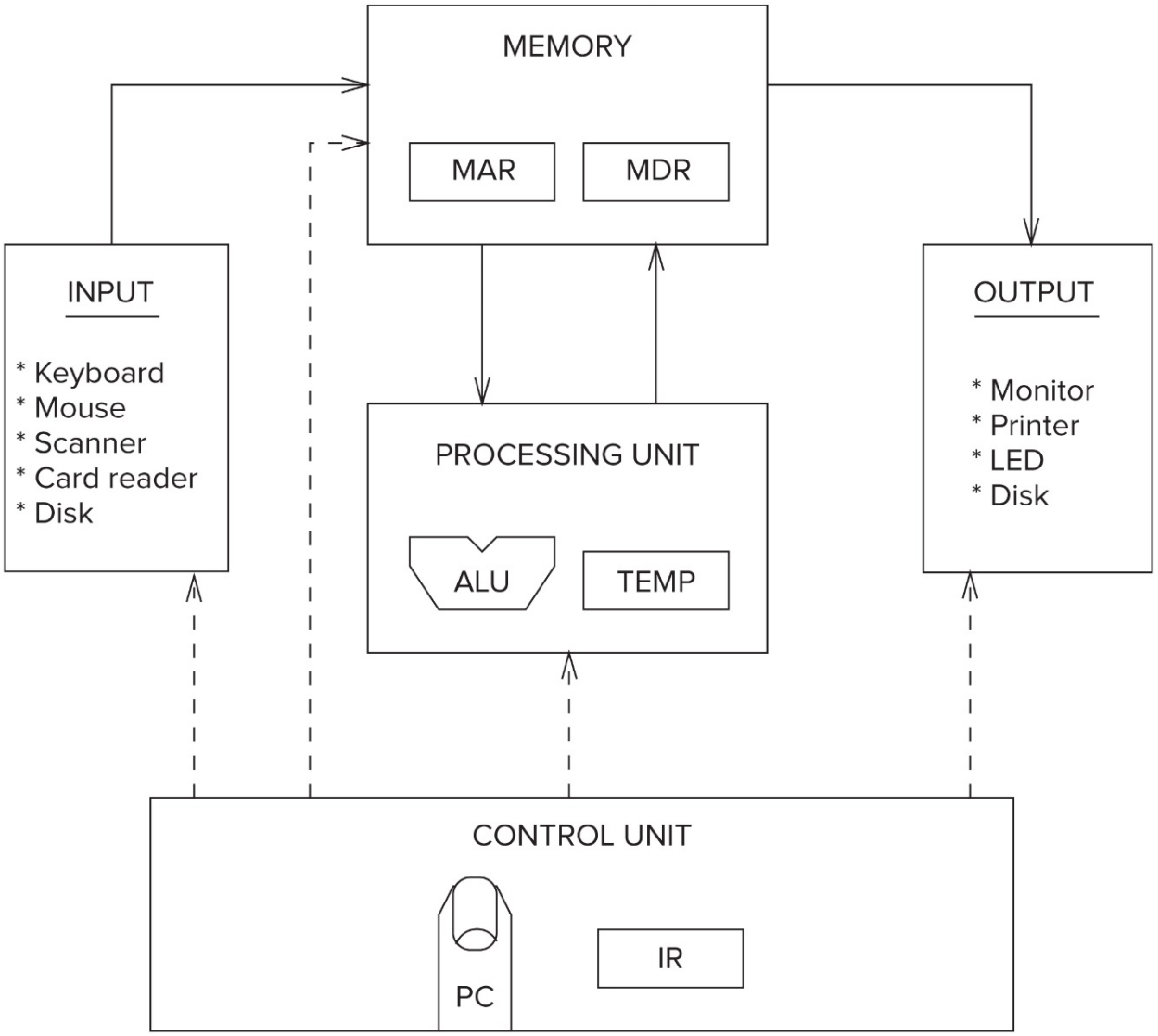
Chapter 4

Solving a Problem using a Computer





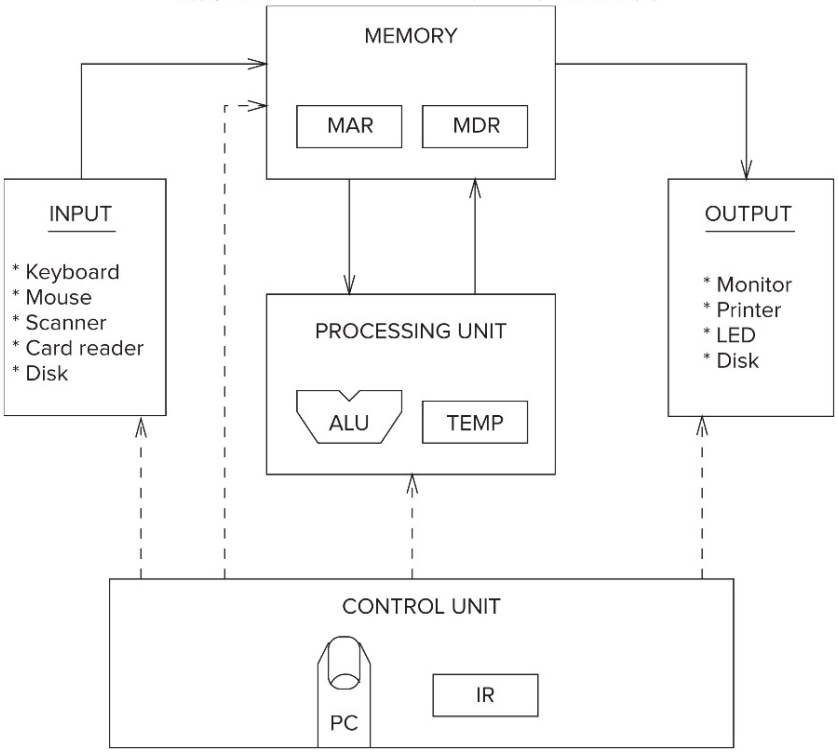
Computer Organization: von Neumann Model



Fundamental model of a computer proposed by John von Neumann in 1946

Computer Organization: von Neumann Model

- Memory: stores data and instructions during program execution
- Processing Unit: transforms data -- logic circuits, temporary storage
- Control Unit: orchestrates fetching of instructions (from memory) and execution of instructions (in processing unit)
- Input: receives data from external environment
- Output: sends data to external environment



Processing Unit + Control Unit = CPU: Central Processing Unit.

Memory

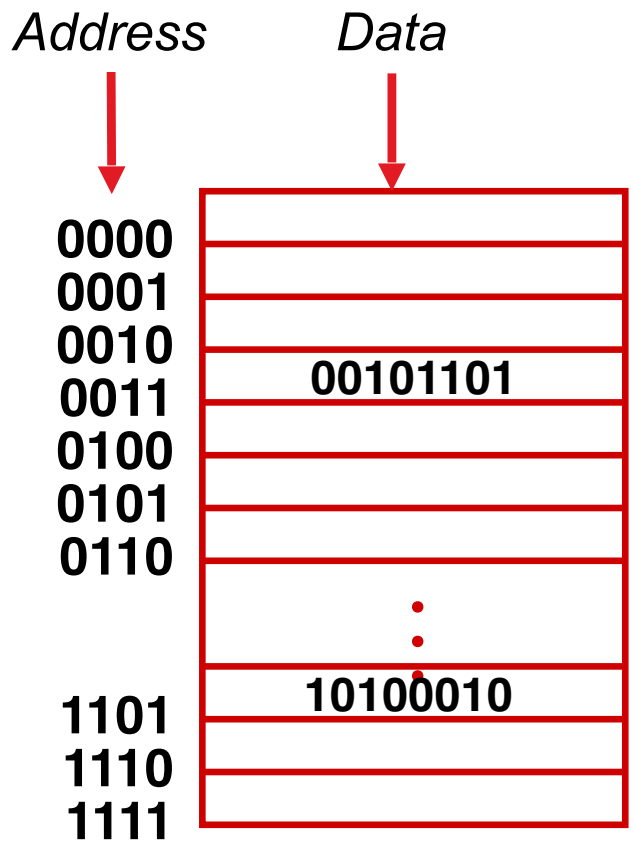
Address space = number of locations

2^n locations means n-bit address.

Addressability = number of bits in each location

Basic operations

- LOAD data from memory to CPU
- STORE data from CPU to memory

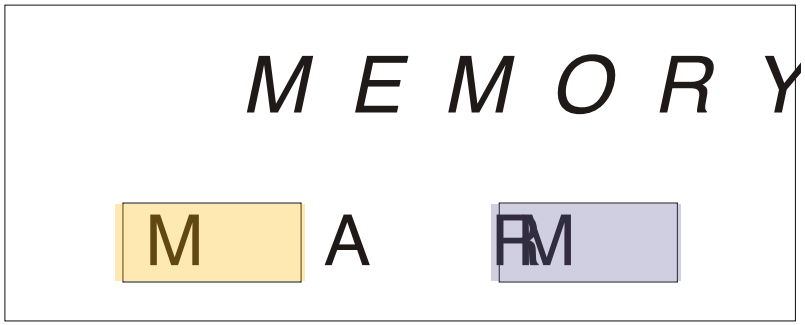


Interface to Memory

Registers used to move data into and out of memory

MAR = Memory Address Register

MDR = Memory Data Register



To LOAD data from memory:

1. Write the address into the MAR
2. Send a "read" signal to memory
3. Read data from MDR

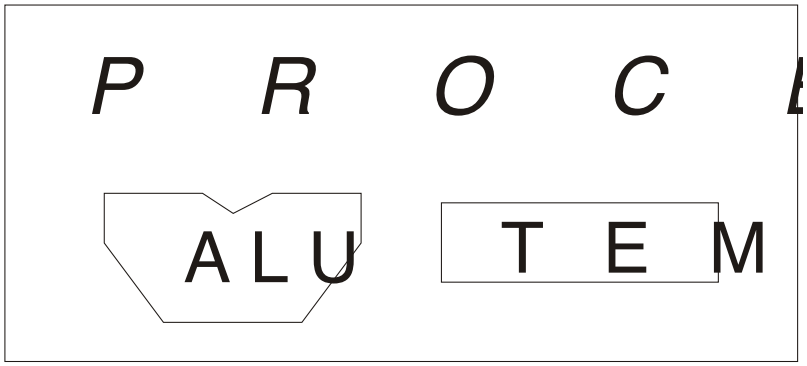
To STORE data to memory:

1. Write the address into the MAR
2. Write the data into MDR
3. Send a "write" signal to memory

Processing Unit

Functional Units

- Perform data transformations
- ALU = Arithmetic / Logic Unit
  add, subtract, AND, ...



Registers

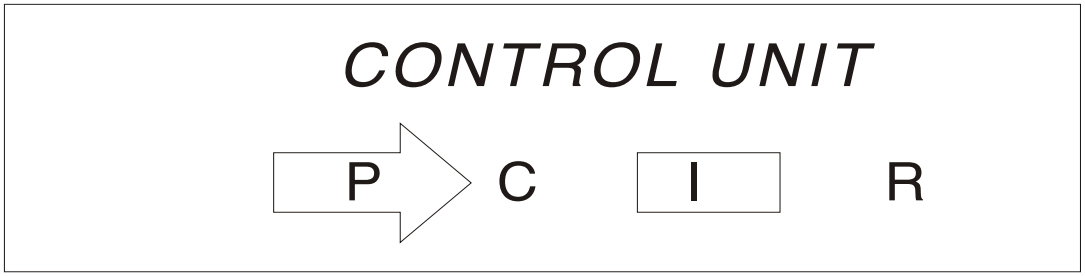
- Temporary storage of data

Word Length

- ALU operations are usually performed on words
- Typical word length:
  64 bits (Intel Core), 32-bit (Intel Atom), 16 bits (LC-3)

Control Unit

Orchestrates the rest of the computer to carry out program instructions



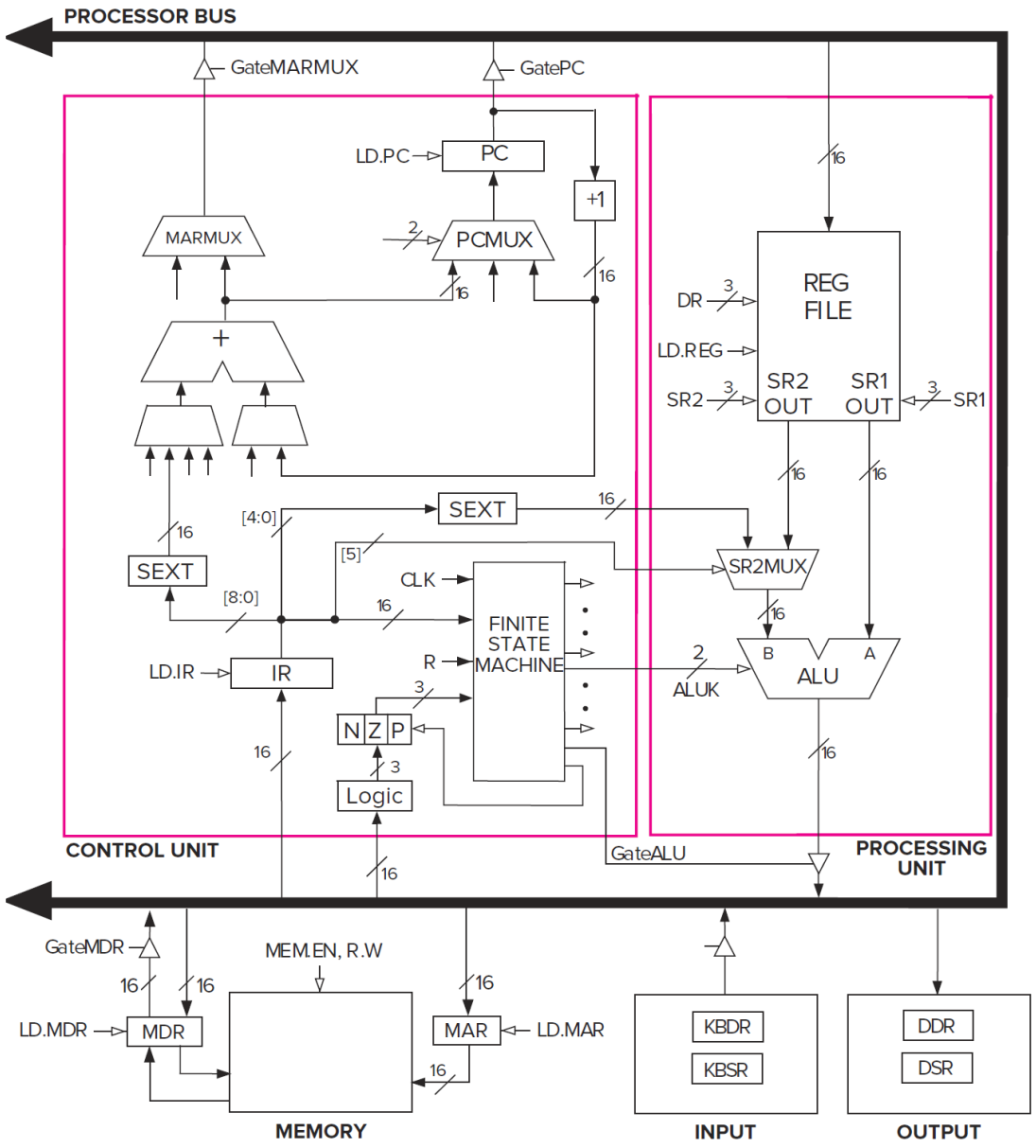
Instruction Register (IR) holds the current instruction.

Program Counter (PC) holds the memory address of the next instruction to be executed. It is known as a "pointer" because it points to a memory location.

Control unit is a state machine that does the following:

1. Read (fetch) the next instruction from memory. (Address is in the PC.)
2. Sends control signals to other parts of processor to move / transform data according to the instruction.
3. Go to step 1, repeat forever...

Example Processor: LC-3



Memory = 2^{16} locations,
16 bits in each location

ALU: performs ADD, AND, NOT
word length is 16 bits

Instruction length is 16 bits

Instruction

A computer **instruction** is a binary value that specifies one operation to be carried out by the hardware. It's binary, so it looks just like the data in Chapter 2 -- a collection of bits. The components of the instruction are *encoded* as various bit fields of the binary value.

Instruction is the fundamental unit of work. Once an instruction begins, it will complete its job.

Two components of an instruction:

opcode specifies the operation to be performed (e.g., ADD)

operands specifies the data to be used during the operation

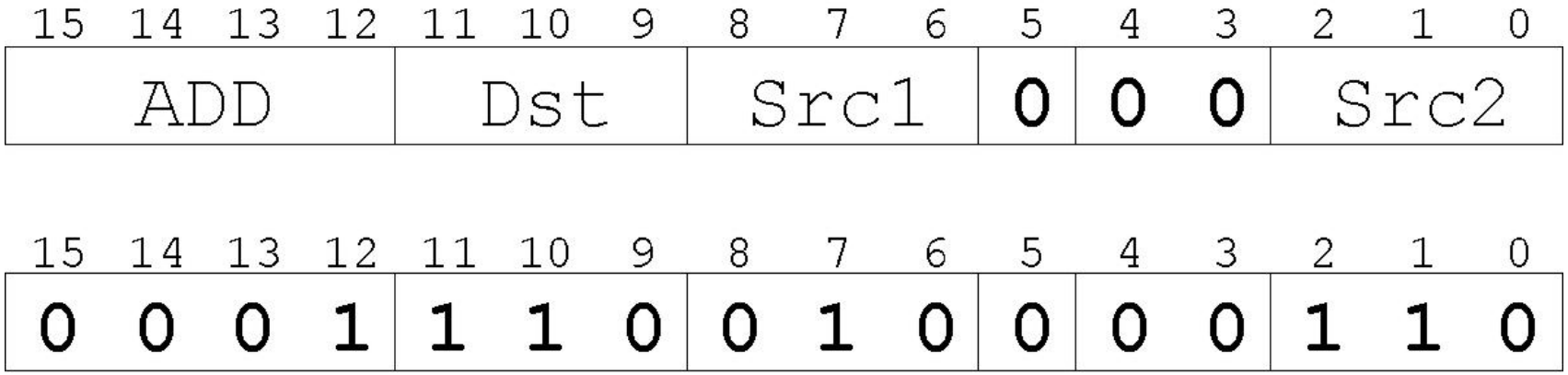
The set of instructions and their formats is called the Instruction Set Architecture (ISA) of a computer.

LC-3 Instruction

LC-3 has a four-bit opcode, bits [15:12] -- up to 16 different operations.

LC-3 has eight registers in the CPU. Registers are used as operands for instructions -- each operand requires 3 bits to specify which register.

Example: ADD instruction, opcode = 0001



"Add the contents of R2 (010) to the contents of R6 (110), and store the result in R6 (110)."

Types of Instructions

Instructions are classified into different types, depending on what sort of operation is specified.

Operate

- Will perform some sort of data transformation
- Examples: add, AND, NOT, multiply, shift

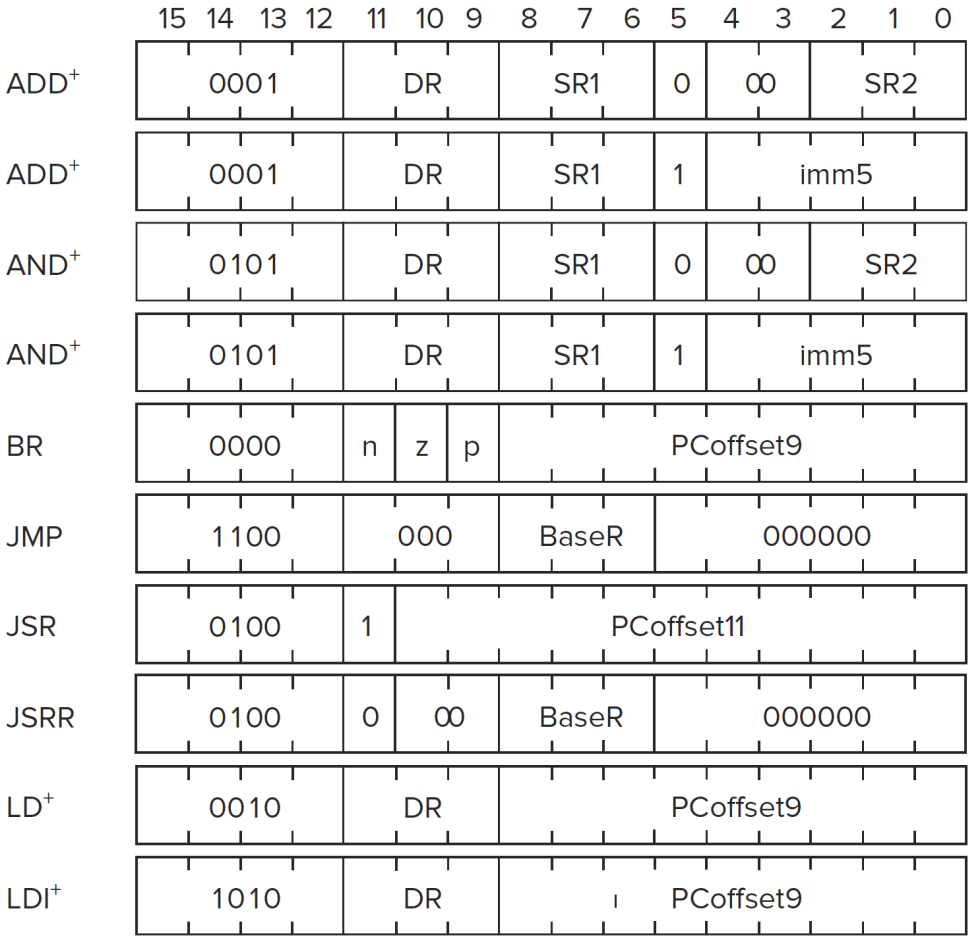
Data Movement

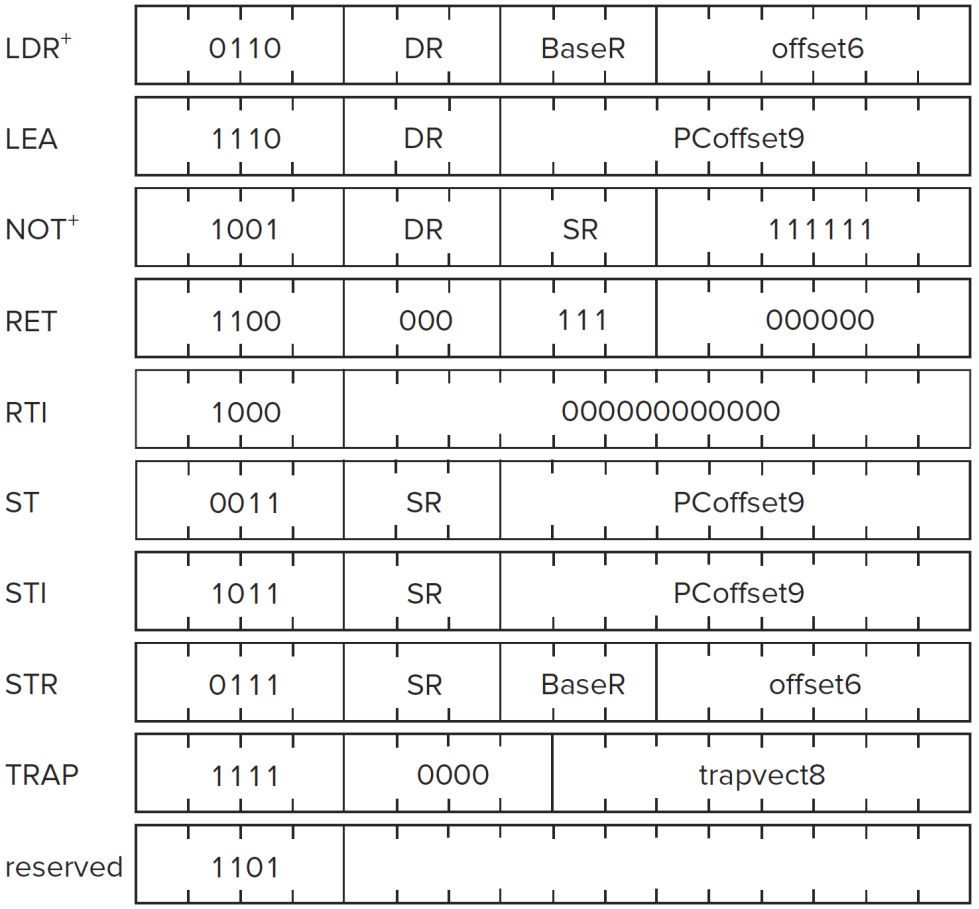
- Move data from memory to a register in the CPU, or
- Move data from a register to a memory location

Control

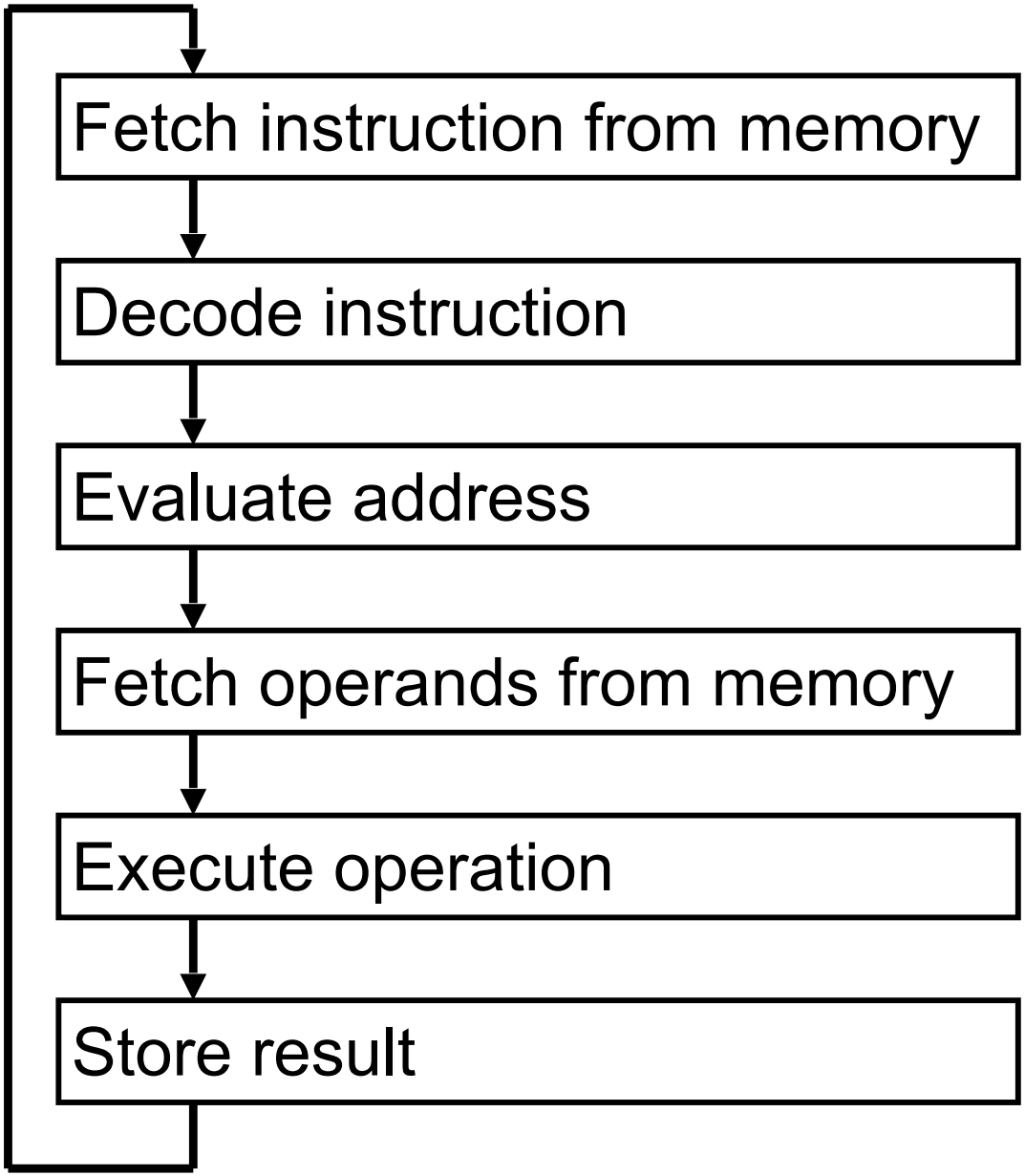
- Determine the next instruction to be executed

LC-3 Instruction Set





Processing an Instruction



The diagram shows the instruction cycle -- the set of steps that must be performed to execute each instruction.

The Control Unit is responsible for performing this sequence of actions.

Each step may require one more more clock cycles.

Instruction Processing: Fetch

Fetch

- PC contains the address of the instruction to be fetched.
- The control unit writes the PC value into the MAR to read the instruction stored at that location.
- It also increments the PC to point to the next consecutive memory location. (It is assumed that the next instruction is stored in the next location in memory. This could be changed when the instruction is executed -- see "control instructions" later in this chapter.)
- When the instruction is retrieved from memory (via the MDR), it is copied into the instruction register (IR).

Instruction Processing: Decode

Decode

- Once the instruction is in the IR, the control unit "looks at" the opcode to determine what operation should be performed.
- One possibility: a 4-bit decoder sets a particular control signal to 1, corresponding to the desired operation.
- The opcode also determines how the other instruction bits should be interpreted.
  - E.g., for ADD, bits [8:6] specify a register to be used as the first operand of the addition.

Instruction Processing: Executing the Instruction

Evaluate Address

For memory instructions, determine which address to load/store.

Fetch Operands

Get data values from registers and/or memory.

Execute

Perform the necessary operations (e.g., add).

Store Result

Write the result of the computation to a register or to memory.

Not every step is required for every instruction. For example, if there is no memory access, then the Evaluate Address step is not performed.

Changing the Sequence of Instructions

In the Fetch step, we incremented the PC to point to the next instruction.

Sometimes, we want to execute a different instruction, not the next one in memory. This is the job of a **control** instruction.

A control instruction may change the PC value. In other words, instead of pointing to the next place in memory, a new address can be calculated and stored in the PC register.

This change may be conditional -- e.g., if the last value calculated was zero, write a new value to the PC. These are typically called branch instructions.

The change may be unconditional -- it always happens. These instructions are usually called jump instructions.

LC-3 control instructions will be explained in Chapter 5.

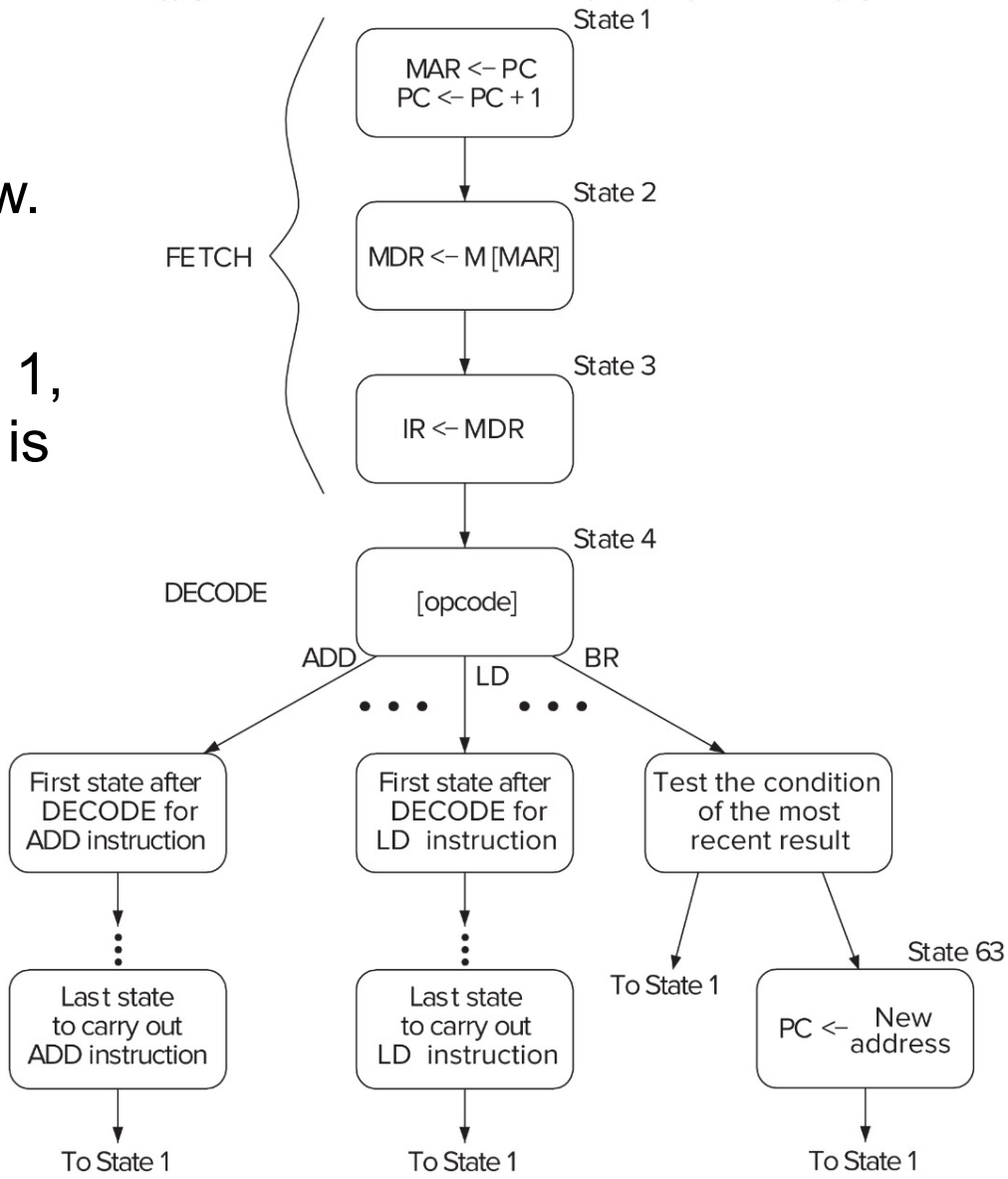
Control of the Instruction Cycle

The control unit is a state machine.
A partial state diagram is shown below.

The instruction cycle starts with State 1, which initiates the Fetch phase. This is done for every instruction.

After the Decode phase (State 4), the states will depend on the specific opcode.

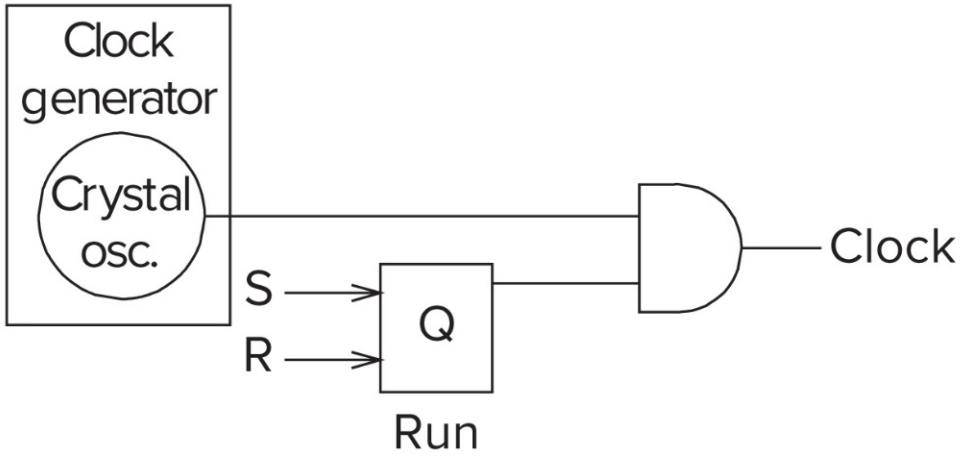
Appendix C has a complete description of the LC-3 state machine.



Stopping Execution

The control unit's state machines keeps running, as long as the clock signal keeps going up and down.

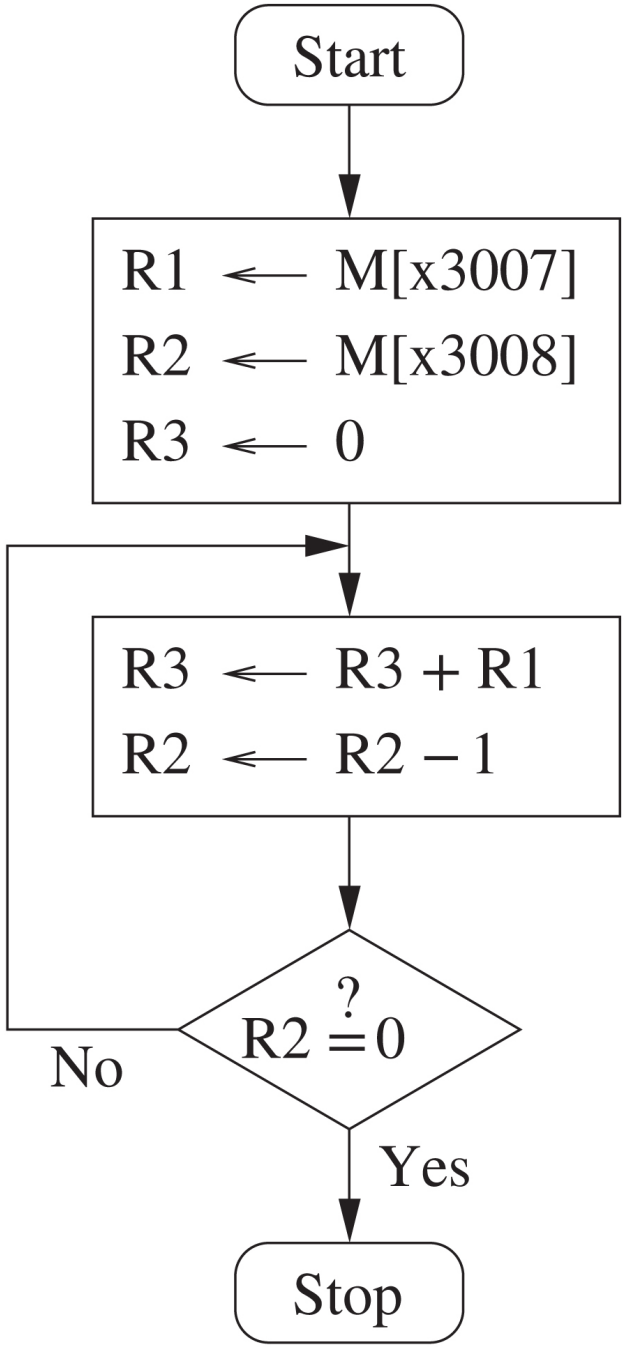
The program can halt execution by turning off the clock signal.



In the circuit above, we "gate" the clock signal using a bit stored in the latch. If the control unit sets that bit to zero, the clock will stop.

In some machines, this is done by a HALT instruction. In LC-3 (and others), the latch is part of a special register controlled by the operating system.

Multiplication Algorithm



Multiplication Algorithm

Address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
x3000	0	0	1	0	0	0	1	0	0	0	0	0	0	1	1	0	R1 <- M[x3007]
x3001	0	0	1	0	0	1	0	0	0	0	0	0	0	1	1	0	R2 <- M[x3008]
x3002	0	1	0	1	0	1	1	0	1	1	1	0	0	0	0	0	R3 <- 0
x3003	0	0	0	1	0	1	1	0	1	1	0	0	0	0	0	1	R3 <- R3+R1
x3004	0	0	0	1	0	1	0	0	1	0	1	1	1	1	1	1	R2 <- R2-1
x3005	0	0	0	0	1	0	1	1	1	1	1	1	1	1	0	1	BR not-zero M[x3003]
x3006	1	1	1	1	0	0	0	0	0	0	1	0	0	1	0	1	HALT
x3007	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	The value 5
x3008	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	The value 4

Table for Question 4.5

Address	Data
0000	0001 1110 0100 0011
0001	1111 0000 0010 0101
0010	0110 1111 0000 0001
0011	0000 0000 0000 0000
0100	0000 0000 0110 0101
0101	0000 0000 0000 0110
0110	1111 1110 1101 0011
0111	0000 0110 1101 1001

Chapter Summary

Computer program is specified by a sequence of instructions, stored in the computer's memory.

Each instruction is a bit pattern that specifies an operation to be performed. Different processors encode this information in different ways.

Combinational and sequential logic is used to retrieve instructions and data from memory, transform data according to the instructions, and write the results back to memory.

The control unit is a state machine, driven by a clock signal. It uses the Program Counter (PC) to step through the instructions in memory, fetching and executing each instruction.

All of this machinery is implemented using the digital logic structures discussed in Chapter 3.

Because learning changes everything.®

www.mheducation.com