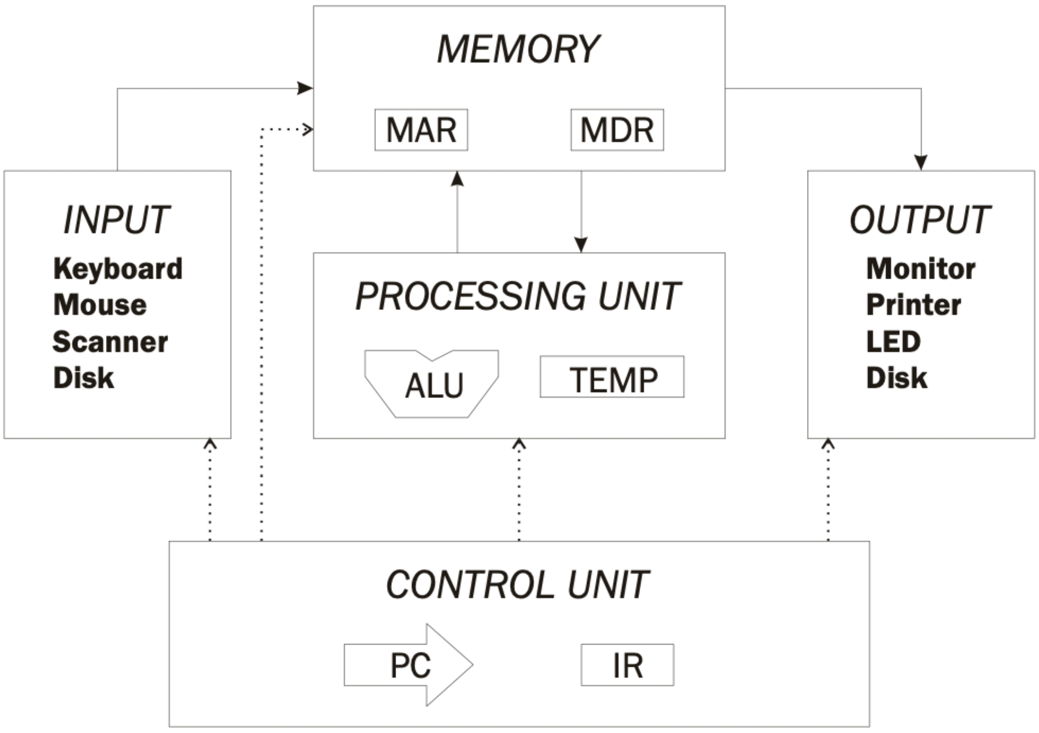
Patt-Ch4 The Von Neumann Model

4.0 Central Ideas

The central idea in the von Neumann model of computer processing is that:

- **the program and data** are both stored as sequences of bits in the computer's memory
- the program is executed **one instruction at a time** under the **direction of the control unit.**

4.1 Basic Components



Overview of Von Neumann Model

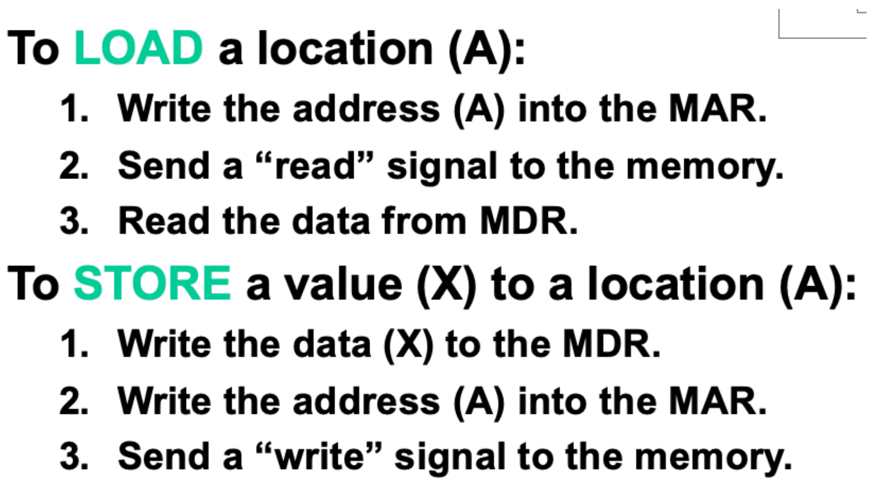
4.1.1 Memory

> both computer programs and data are contained in memory

- **2 Characteristics of Memory**
  - Address
  - Content(Data)
- **2 Registers**
  - **MAR:** Memory Address Register
  - **MDR:** Memory Data Register

MEMORY
MAR
MDR

- **2 Operations**
  - LOAD: read
  - STORE: write


To LOAD a location (A):

1. Write the address (A) into the MAR.
2. Send a “read” signal to the memory.
3. Read the data from MDR.

To STORE a value (X) to a location (A):

1. Write the data (X) to the MDR.
2. Write the address (A) into the MAR.
3. Send a “write” signal to the memory.

4.1.2 Processing Unit

Functional Units

- could have many functional units.Some of them special-purpose
  - e.g. multiply, square root, etc.
- **ALU** (*Arithmetic and Logic Unit*) is the simplest function unit
  - e.g. ADD,SUBSTRACT(*Arithmetic*), AND,OR,NOT(*Logic*)
  - LC-3's ALU performs ADD,AND,NOT

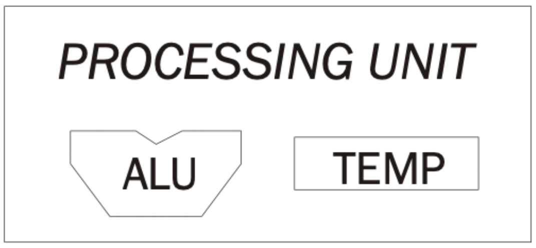
Word and Word Length

- ***instruction***: the smallest piece of work for computer to carry out
- ***word length(size)***: the **fixed size**(bits) of the data elements processed by **ALU** in one instruction
  - In short(maybe not correct), **the number of bits of one instruction**
  - usually also **width of registers**
- ***word***: the data elements to be processed
- each ISA has its own word length, depending on the intended use of the computer
  - most microprocessors today in PC,even in cell phones, have a word length of 64 bits
  - the word length for LC-3 is 16 bits, LC-3 is *word accessible*

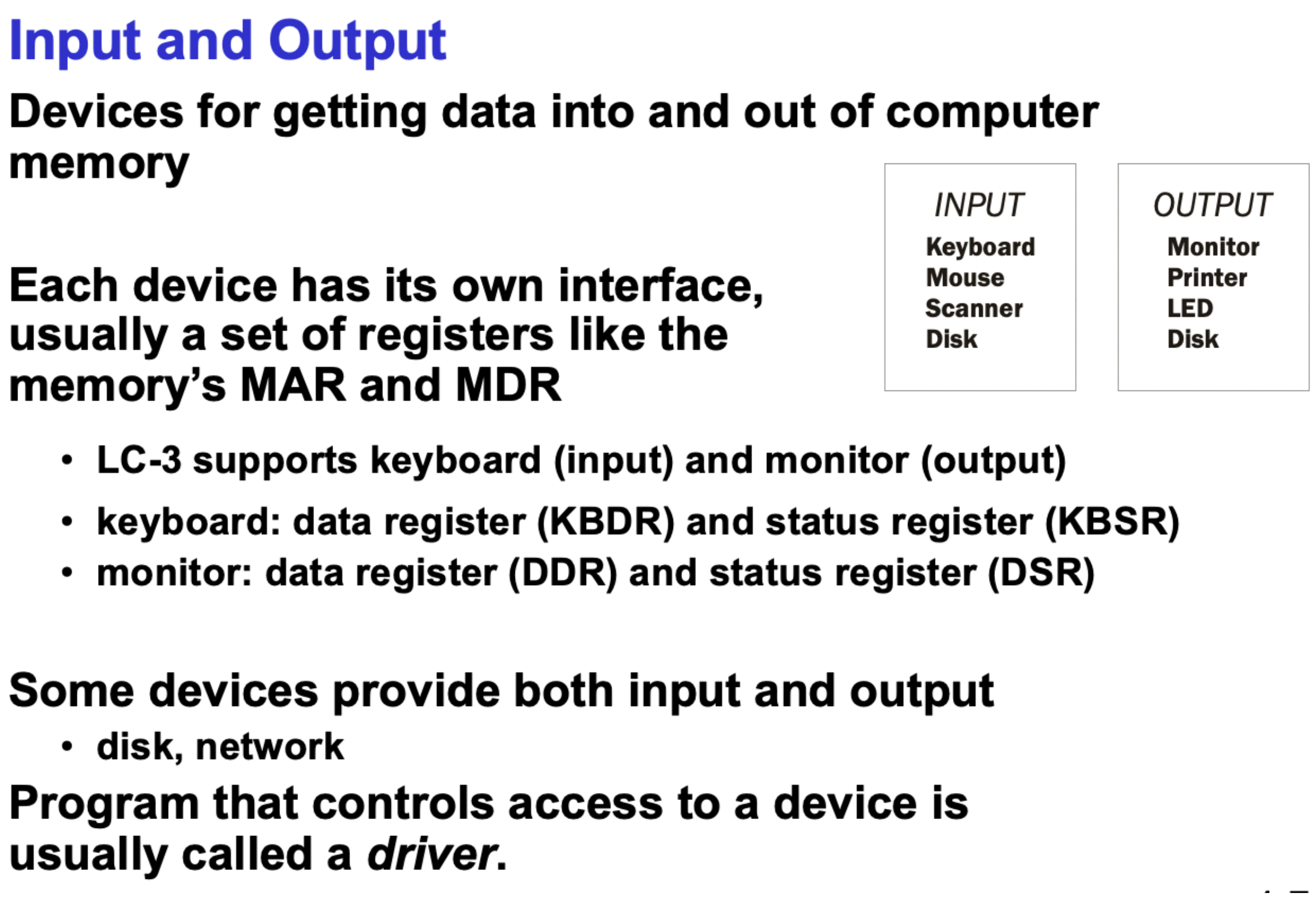
Registers

- small, temporary storage
- store operands and results of functional units
  - the purpose is to save time for accessing memory,which is TOO SLOW compared to ALU computations
- Current microprocessors typically contain 32 registers, each consisting of 32 or 64 bits, depending on the architecture.

- LC-3 has eight registers (R0, ..., R7), each 16 bits. 8 registers need 3bits to identify



4.1.3 Input and Output



Input and Output

Devices for getting data into and out of computer memory

Each device has its own interface, usually a set of registers like the memory's MAR and MDR

- **LC-3 supports keyboard (input) and monitor (output)**
- **keyboard: data register (KBDR) and status register (KBSR)**
- **monitor: data register (DDR) and status register (DSR)**

Some devices provide both input and output

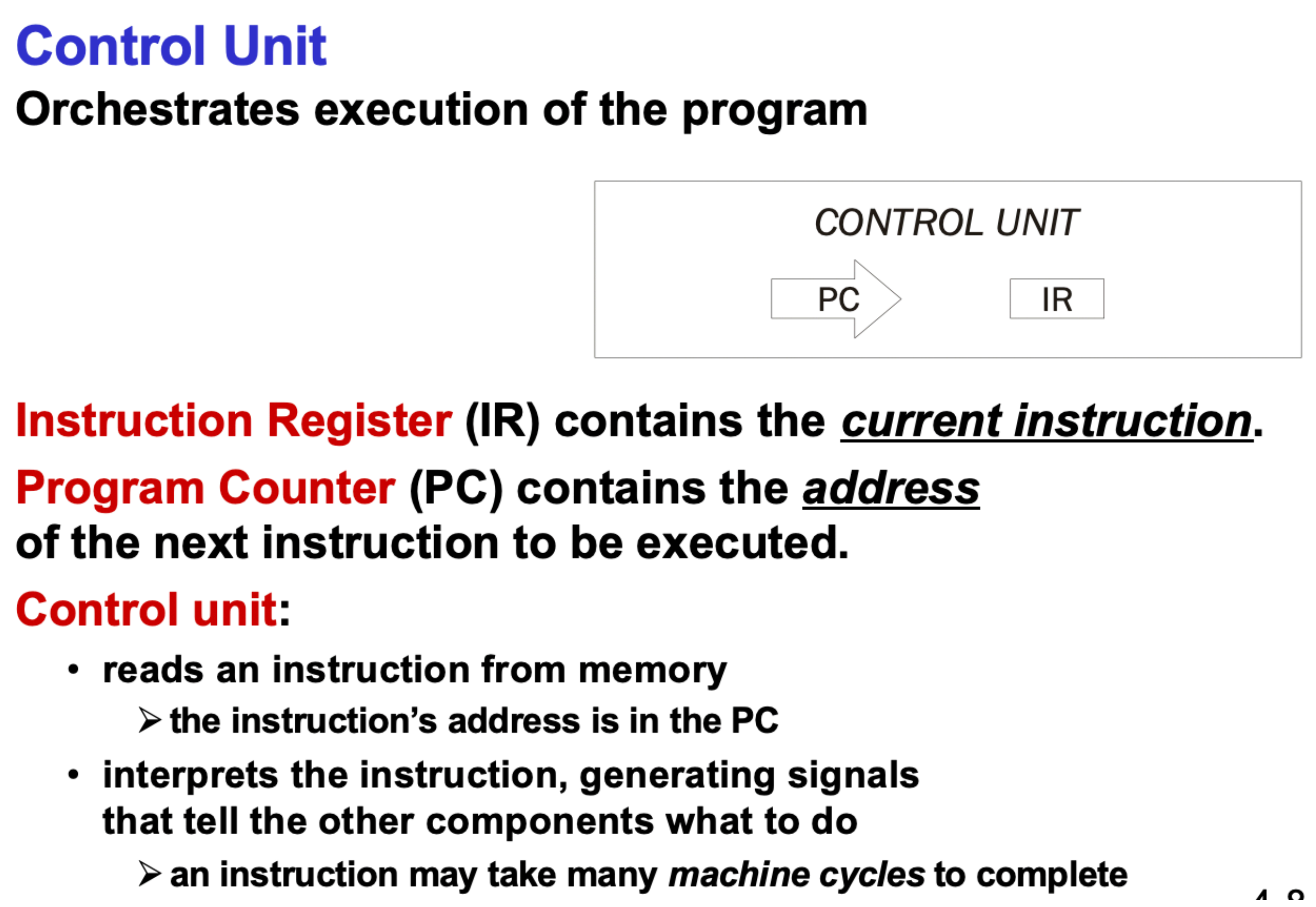
- **disk, network**

Program that controls access to a device is usually called a *driver*.

INPUT
Keyboard
Mouse
Scanner
Disk

OUTPUT
Monitor
Printer
LED
Disk

4.1.4 Control Unit


CONTROL UNIT
PC
IR

Control Unit

Orchestrates execution of the program

Instruction Register (IR) contains the *current instruction*.

Program Counter (PC) contains the *address* of the next instruction to be executed.

Control unit:

- **reads an instruction from memory**
  - ➢ **the instruction's address is in the PC**
- **interprets the instruction, generating signals that tell the other components what to do**
  - ➢ **an instruction may take many *machine cycles* to complete**

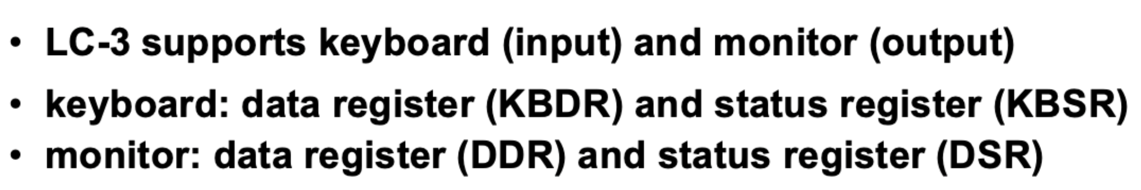
> PC(*Program Counter*) = IP(*Instruction Pointer*)

4.2 LC-3: Example von Neumann Machine

Memory

- MAR for addressing individual locations, MDR for holding the contents of a memory location
- MAR contains 16 bits, since LC-3 address space is 2^{16}
- MDR contains 16 bits, since addressability is 16 bits(*a word for LC-3*)

I/O



- **LC-3 supports keyboard (input) and monitor (output)**
- **keyboard: data register (KBDR) and status register (KBSR)**
- **monitor: data register (DDR) and status register (DSR)**

Processing Unit

- ALU: ADD,AND,NOT
- 8 registers(R0,...,R7)
- word length: 16 bits

Control Unit

- Finite State Machine: the most important structure, which directs all activities

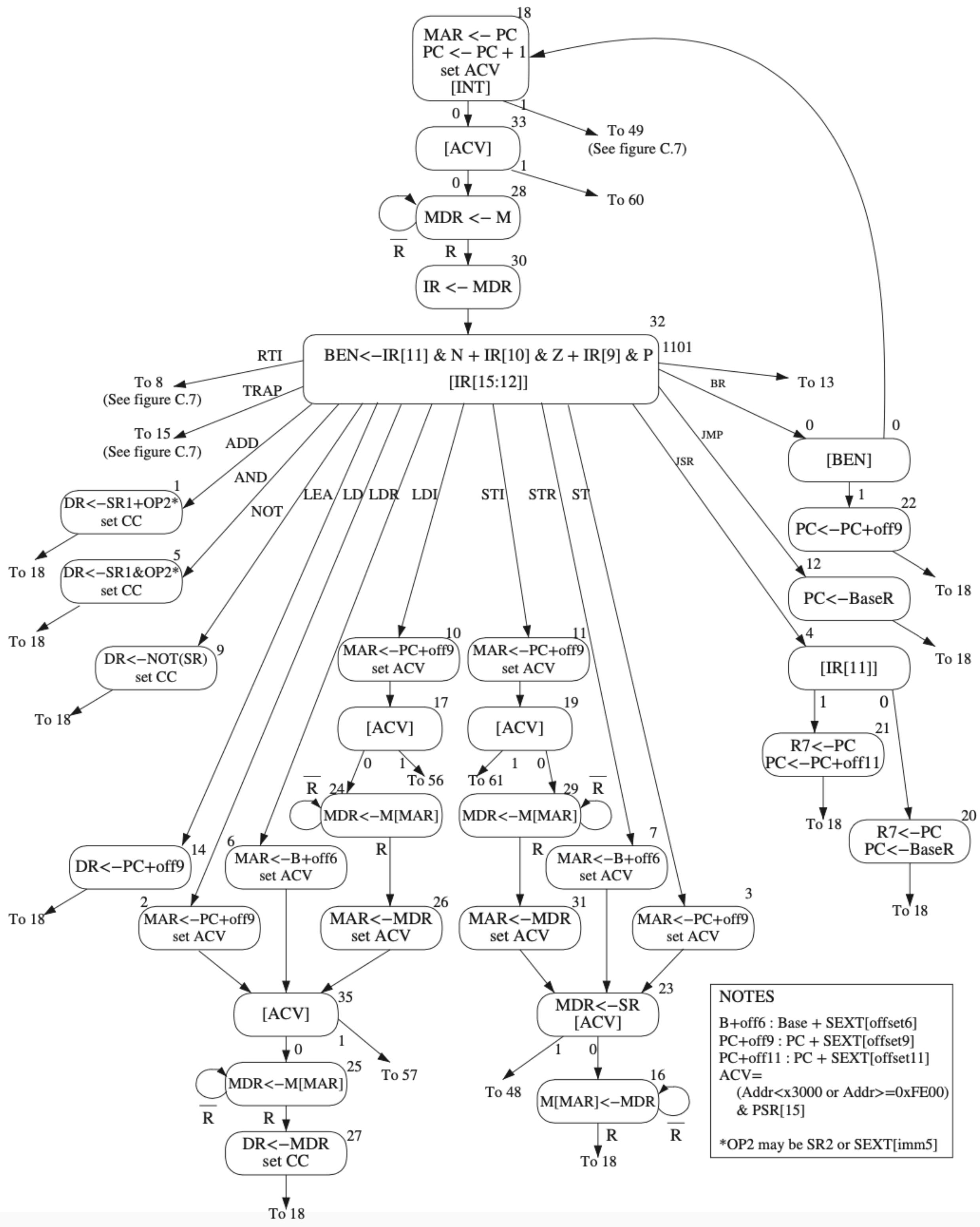


Figure C.2 **A state machine for the LC-3.**

- CLK: specifies how long each clock cycle lasts
- IP
- PC

4.3 Intruction Processing

4.3.1 The Instruction

4.3.1.1 Some Concepts about Instruction

Fundamental Unit

The instruction is the fundamental unit of work.

Specify 2 things

- **opcode**: operation to be performed
- **operands**: data/locations to be used for operation

Storage Form

- encoded as **a sequence of bits**
- often have a fixed word length, such as 16 or 32 bits
- Control Unit interprets instructions, and generate control signals to carry out operations
- Operations is either executed completely, or not at all

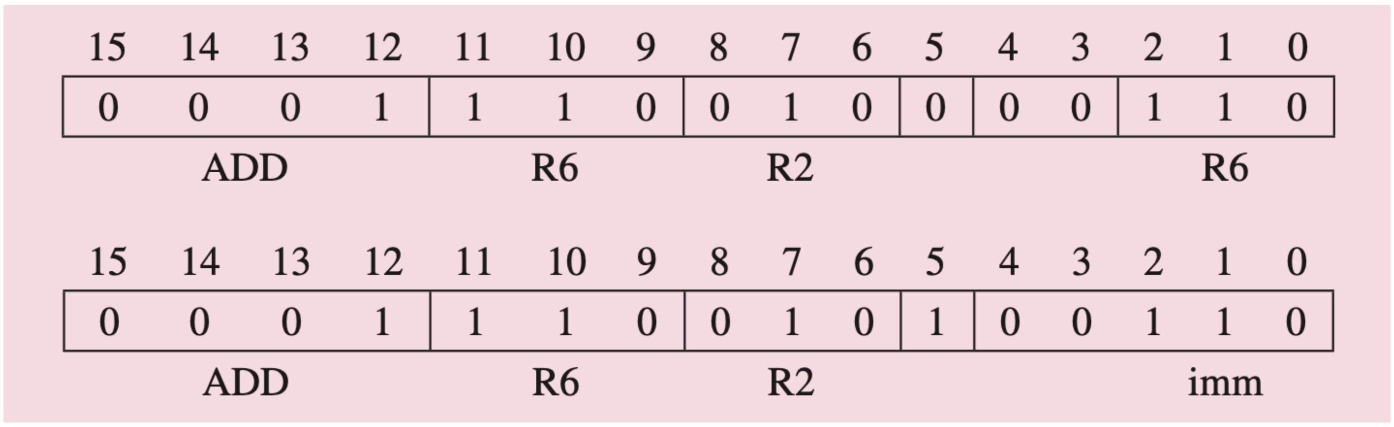
3 Basic Kinds

- **Operate** instructions: operate on data
  - Arithmetic: ADD,SUBSTRACT
  - Logic: AND,OR,NOT
- **Data Movement**: move information, processing unit <-> memory or I/O
  - LD: load
  - ST: store
- **Control** instructions: alter the sequence
  - normally the next instruction executed is the instruction contained in the next memory location
  - TRAP: halt
  - BR: condition

LC-3 Instruction

- 16 bits(one word)
- Bits[15:12] opcode
- Bits[11:0] figure out where the operations are

4.3.1.2 Example: LC-3 ADD Instruction



[15:12] ADD *opcode*

[11:9] the destination register

[8:6] one of the two operands

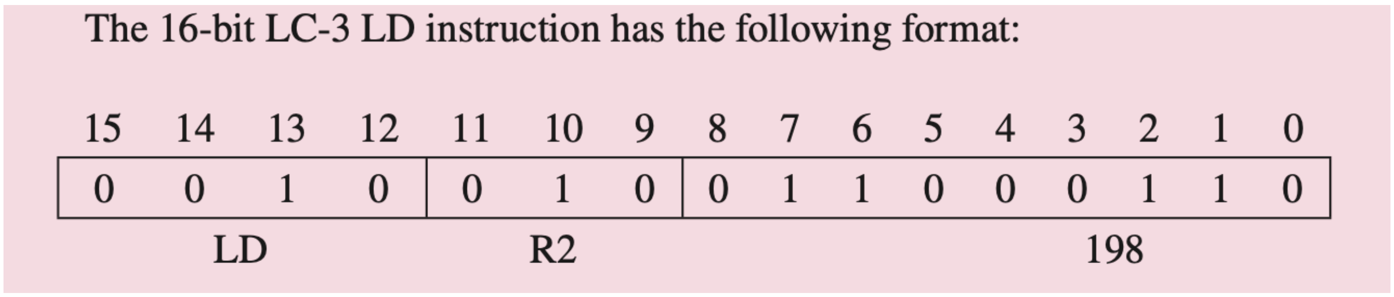
[5] format

[4:0] either a immediate or the other register

4.3.1.3 Example: LC-3 LD Instruction

> LD, stands for load

- ***Addressing Mode***: formulas used for calculating the address of the memory location to be read
- **PC + offset** is a particular addressing mode
  - sign-extending the 2's complement integer contained in bits [8:0] to 16 bits
  - adding it to the current contents of PC *(Noting:PC has been already +1)*


The 16-bit LC-3 LD instruction has the following format:

add 198 to the contents of the PC to form the address of a memory location

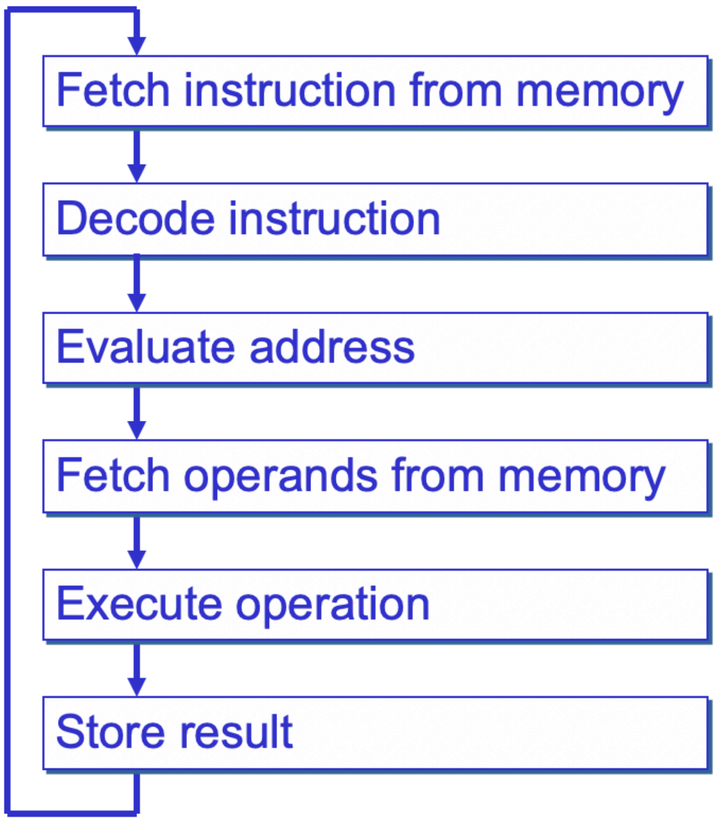
load the contents of that memory location into R2.

4.3.2 The Instruction Cycle

- ***Instruction Cycle***: The entire sequence of steps needed to process an instruction
- consists of **six sequential *phases***, each phase requiring zero or more steps (clock cycle).
- zero means not all instructions require all six phases.
  - ADD: FETCH -> DECODE -> FETCH OPERANDS -> EXECUTE, excluding EVALUATE ADRESS and

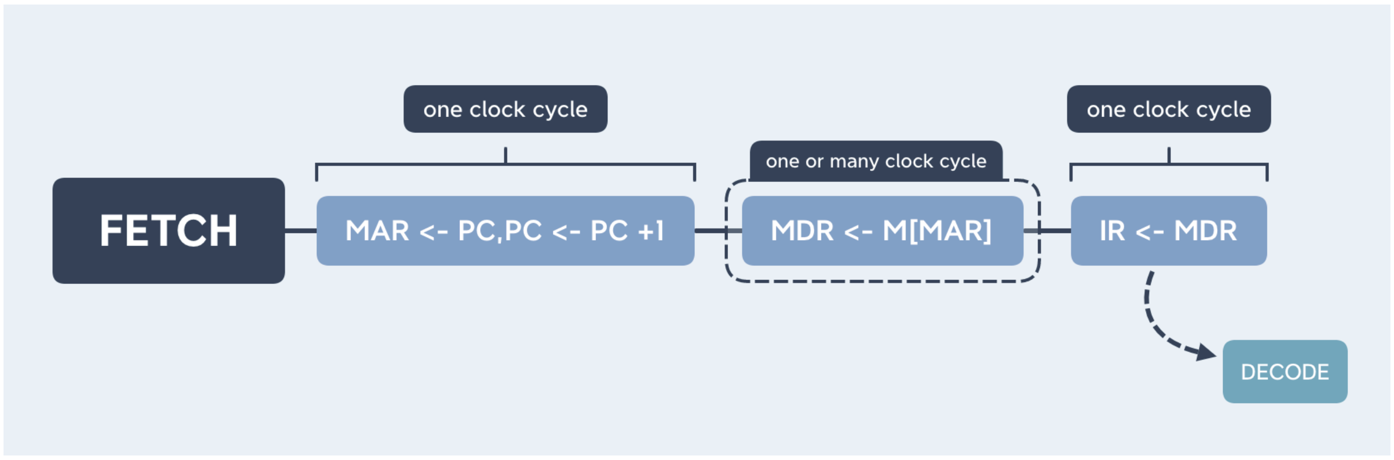
STORE RESULT

- LD: don't need EXECUTE

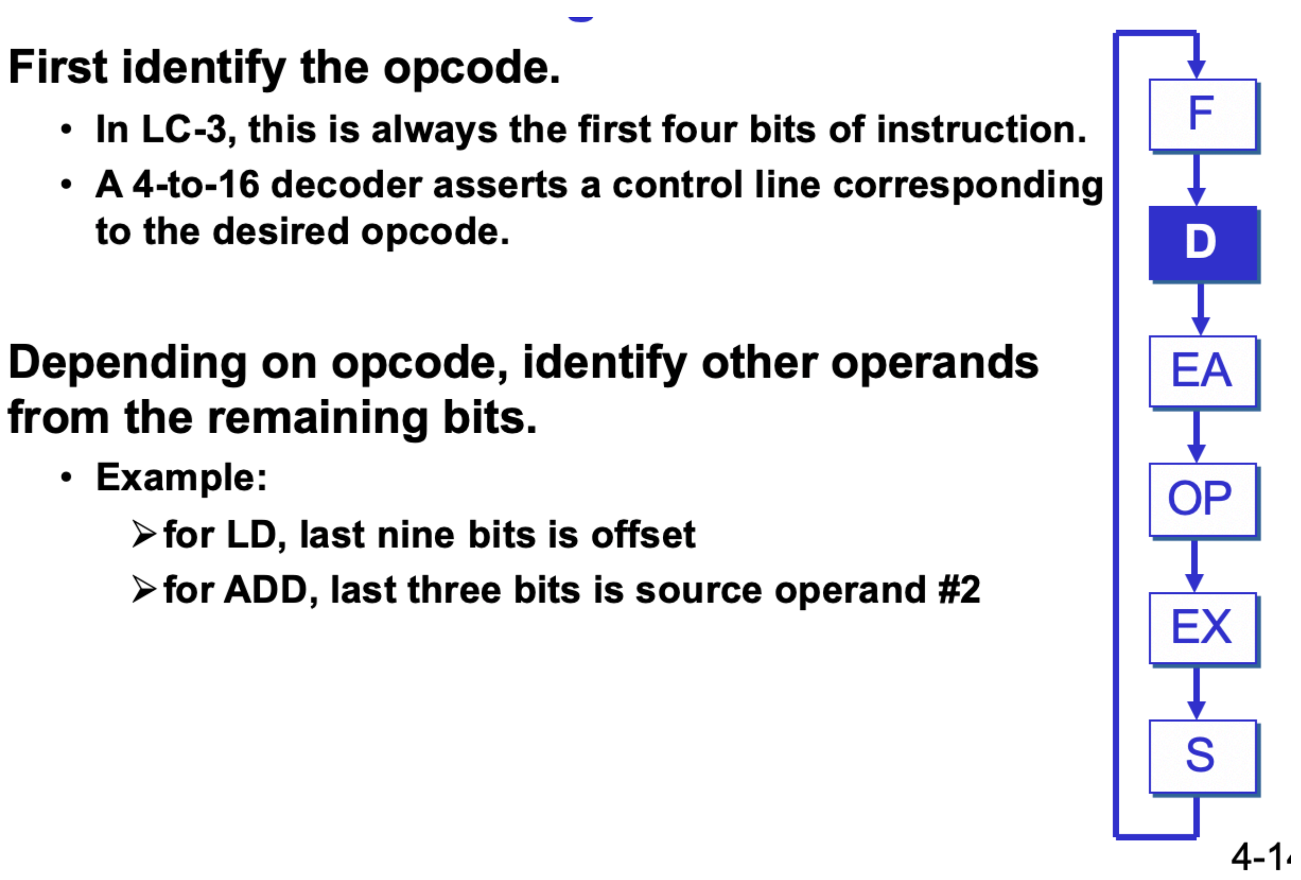


4.3.2.1 FETCH*

> machine cycle = clock cycle



4.3.2.2 DECODE*



First identify the opcode.

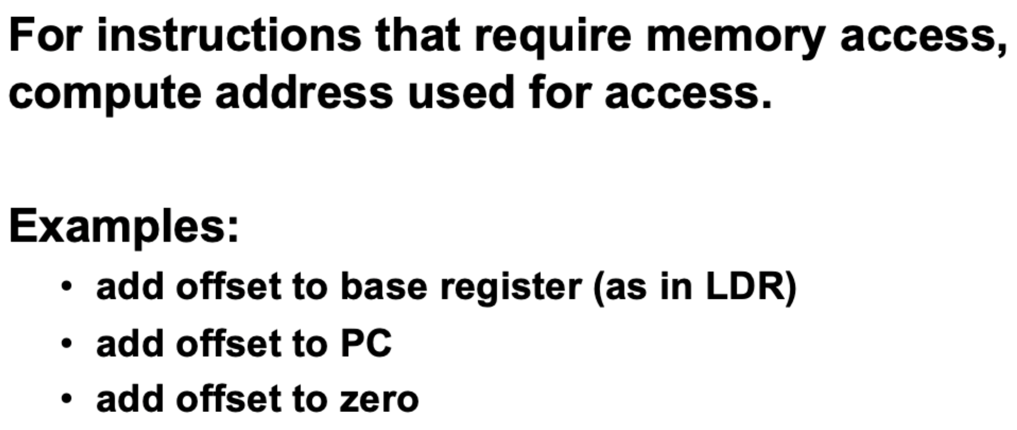
- **In LC-3, this is always the first four bits of instruction.**
- **A 4-to-16 decoder asserts a control line corresponding to the desired opcode.**

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

- **Example:**
  - **for LD, last nine bits is offset**
  - **for ADD, last three bits is source operand #2**

4-1

4.3.2.3 EVALUATE ADDRESS



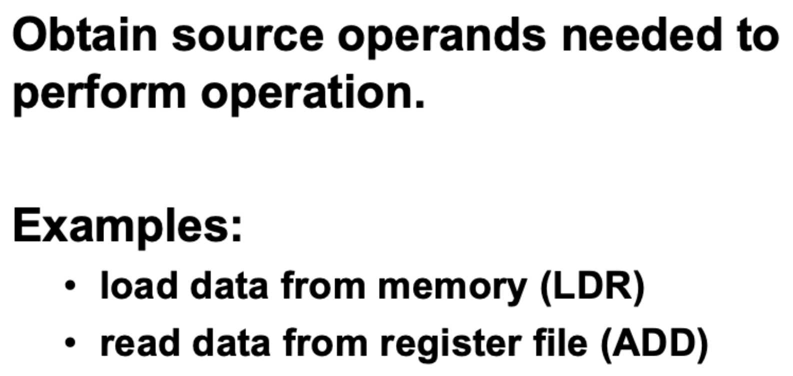
For instructions that require memory access, compute address used for access.

Examples:

- **add offset to base register (as in LDR)**
- **add offset to PC**
- **add offset to zero**

- Not all instructions access memory to load or store data.
- For example, the ADD and AND instructions in the LC-3 obtain their source operands from **registers** or from the **instruction itself** and store the result of the ADD or AND instruction in a register.

4.3.2.4 FETCH OPERANDS

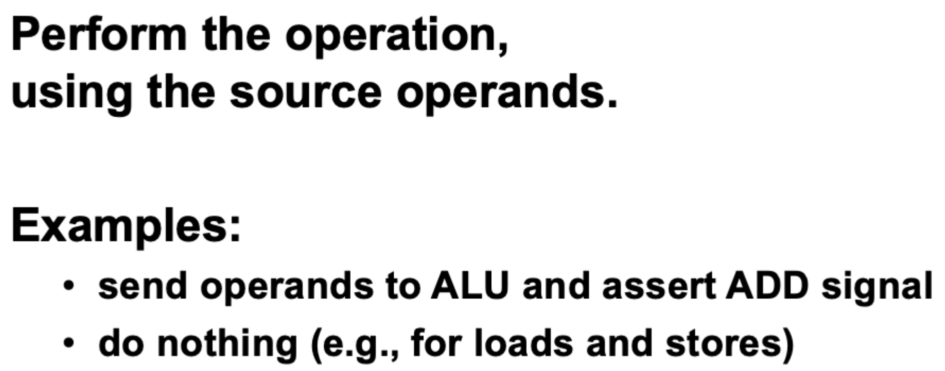


Obtain source operands needed to perform operation.

Examples:

- load data from memory (LDR)
- read data from register file (ADD)

4.3.2.5 EXECUTE

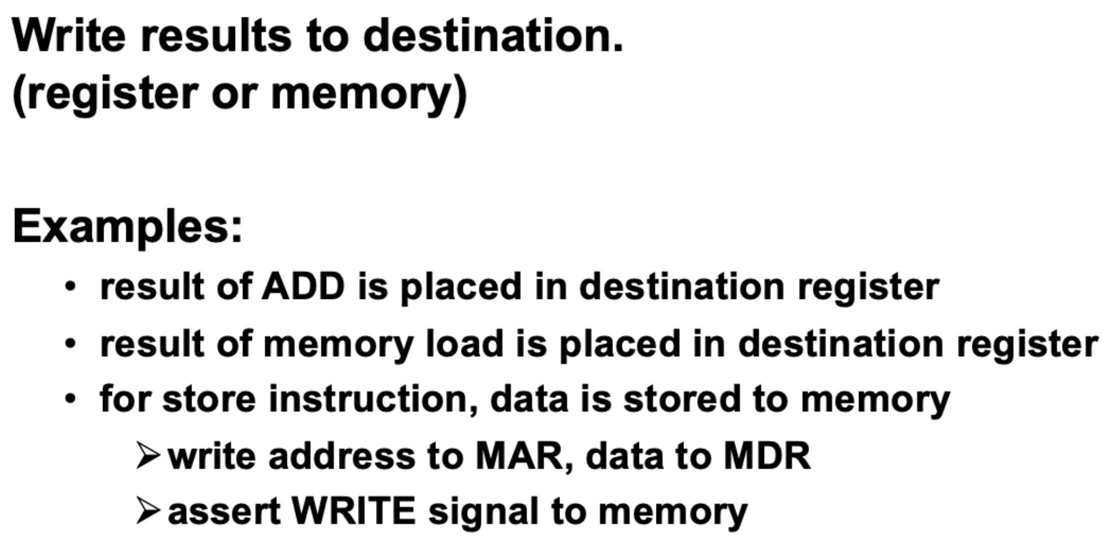


Perform the operation, using the source operands.

Examples:

- send operands to ALU and assert ADD signal
- do nothing (e.g., for loads and stores)

4.3.2.6 STORE RESULT



Write results to destination. (register or memory)

Examples:

- result of ADD is placed in destination register
- result of memory load is placed in destination register
- for store instruction, data is stored to memory
  - ➢ write address to MAR, data to MDR
  - ➢ assert WRITE signal to memory

- In the case of the ADD instruction, in many computers this action is performed during the EXECUTE phase.
- That is, in many computers, including the LC-3, an ADD instruction can fetch its source operands, perform the ADD in the ALU, and store the result in the destination register **all in a single clock cycle.**
- A separate STORE RESULT phase **is not needed**.

4.4 First Program: Multiplication

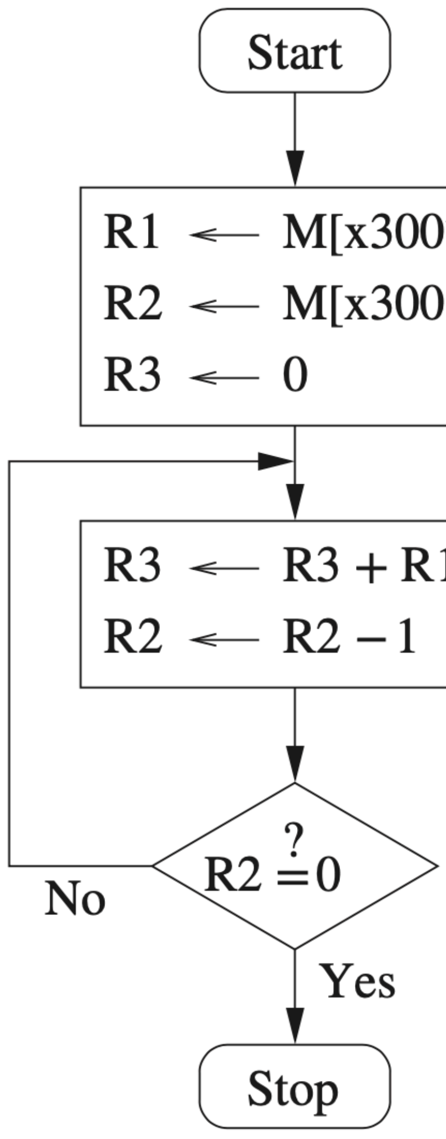


Figure 4.6 Flowchart for an algorithm that multiplies two positive integers.

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`

Figure 4.7 A program that multiplies without a multiply instruction.