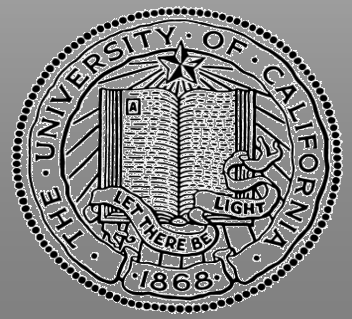
LC-3 Architecture

Textbook Chapters 4-5



Real Programmers Do It In Octal

- Seymour Cray, designer of the Cray line of supercomputers, was among the greatest [of the “Real Programmer” culture]. He is said once to have toggled an entire operating system of his own design into a computer of his own design through its front-panel switches. In octal. Without an error. And it worked. Real Programmer macho supremo.
  - Eric S. Raymond, *The Cathedral and the Bazaar*

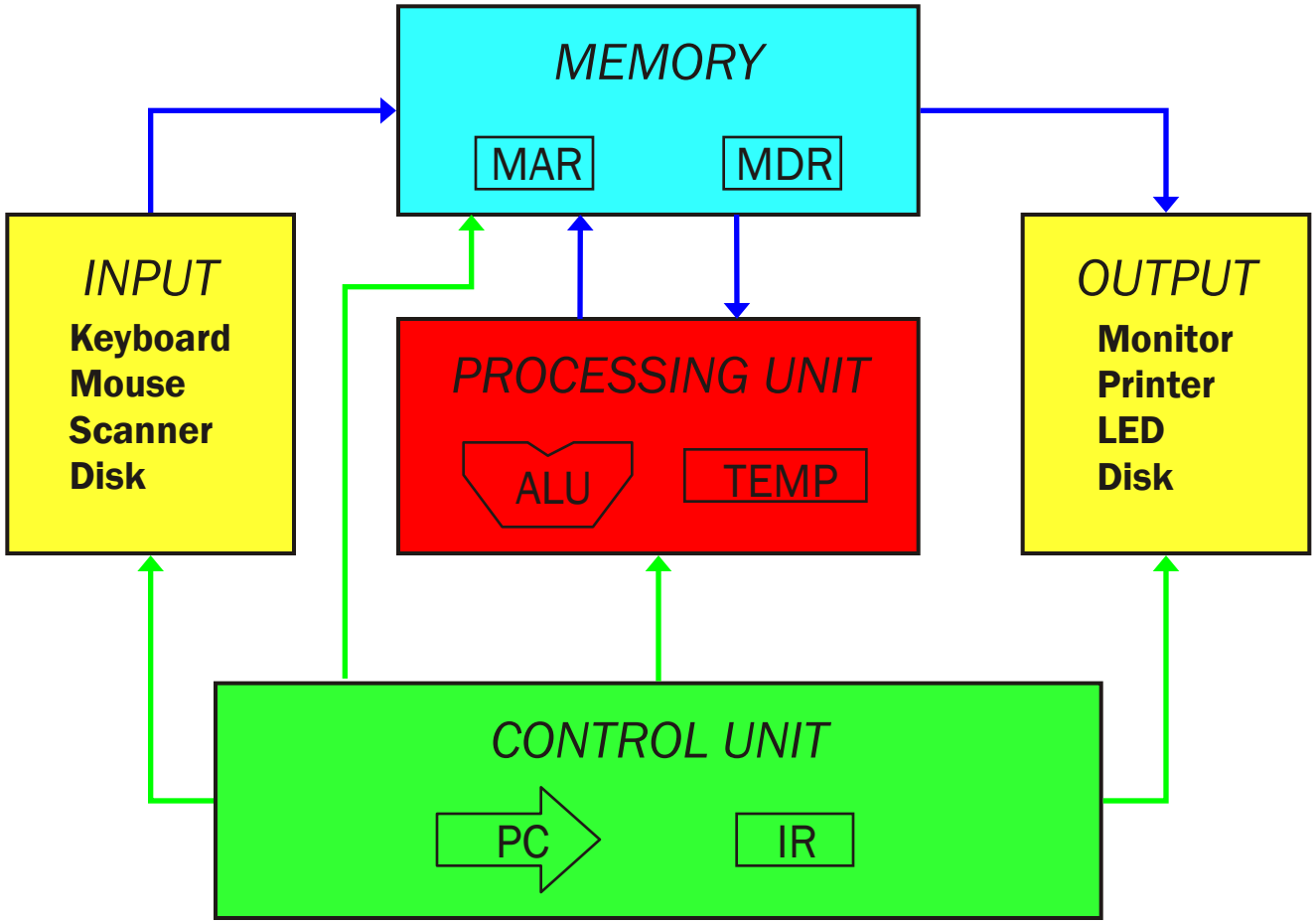
The “Stored-Program” Computer

- 1943: ENIAC
  - Hard-wired program with setting dials and switches
- 1944: Beginnings of EDVAC
  - Electronic Discrete Variable Automatic Computer
  - Program stored in memory
- 1945: John von Neumann
  - Wrote a report on the “stored-program computer”
  - Known as the *First Draft of a Report on the EDVAC*

First Draft of a Report on EDVAC

- The basic structure proposed became known as the “von Neumann machine” (or model)
- This machine had five main components
  1. The *Central Arithmetical* part, CA
  2. The *Central Control* part, CC
  3. The *Memory*, M, for both
  4. Instructions and data
  5. The *Input*, I
  6. The *Output*, O

Von Neumann Model*



* A slightly modified version of Von Neumann's original diagram

CISC vs. RISC

- CISC: Complex Instruction Set Computer
  - Lots of instructions of variable size
  - Memory optimal
  - Fewer registers
- RISC: Reduced Instruction Set Computer
  - Fewer instructions, all of a fixed size
  - More registers
  - Optimized for speed
  - Usually called a load/store architecture

What is “Modern”

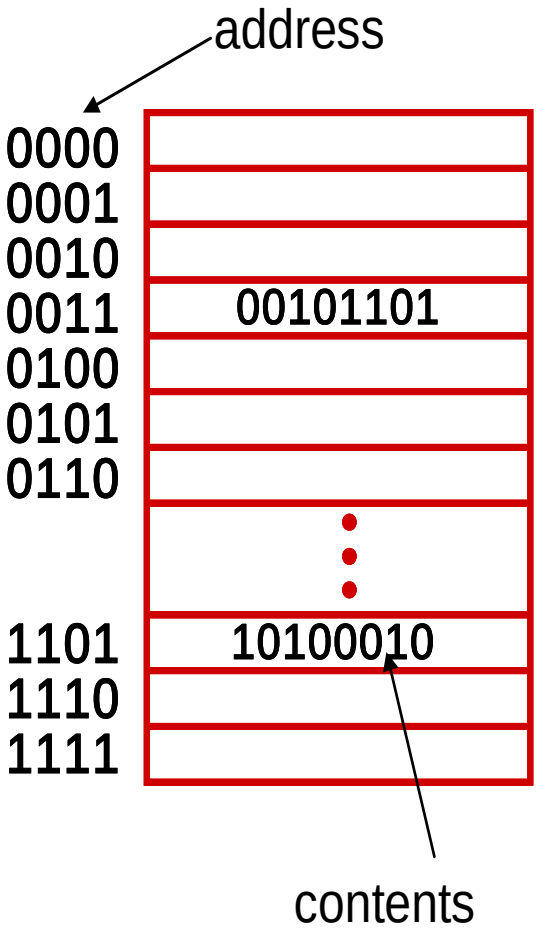
- For embedded applications and for workstations there exist a wide variety of CISC and RISC and CISCy RISC and RISCy CISC
- Most current PCs use the best of both worlds to achieve optimal performance

LC-3 Architecture

- The LC-3 is RISC
  - 15 instructions
- 16-bit data and address (16-bit words)
- 8 general-purpose registers (GPR)
- Plus 4 special-purpose registers
  - Program Counter (PC)
  - Instruction Register (IR)
  - Condition Code Register (CC)
  - Process Status Register (PSR)

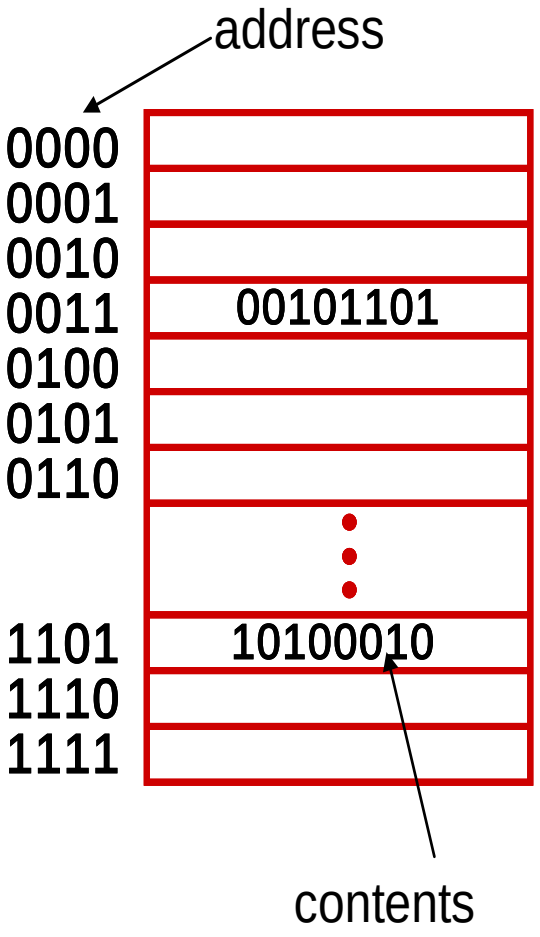
Memory: Address and Contents

- Memory is a 2^k x m array of stored bits
- Address
  - Unique (k-bit) identifier of location
  - LC-3: k = 16
- Contents
  - m-bit value stored in location
  - LC-3: m = 16



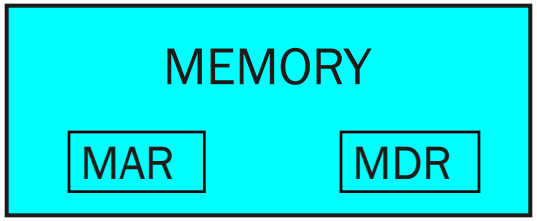
Memory: Basic Operations

- LOAD
  - Read a value from a memory location
- STORE
  - Write a value to a memory location



Interface to Memory

- How does the processing unit get data to/from memory?
  - MAR: Memory Address Register
  - MDR: Memory Data Register



Interface to Memory

- To **LOAD** from a location (A):
  - Write the address (A) into the MAR
  - Send the READ signal to the memory
  - Read the data from MDR
- To **STORE** a value (X) into a location (A):
  - Write the data (X) to the MDR
  - Write the address (A) into the MAR
  - Send a WRITE signal to the memory

Input and Output

- Input devices
  - Keyboard
  - Mouse
  - Scanner
  - …
- Output devices
  - Monitor
  - Printer
  - LEDs
  - …

Input and Output

- Input and output are devices for…
  - Getting data into computer memory
  - Getting data out of computer memory
- Each device has its own interface
  - Usually a set of registers
  - E.g., memory's MAR and MDR
- Some devices provide both input and output
  - E.g., disk drive, network device
- The program that controls access to a device is usually called a driver

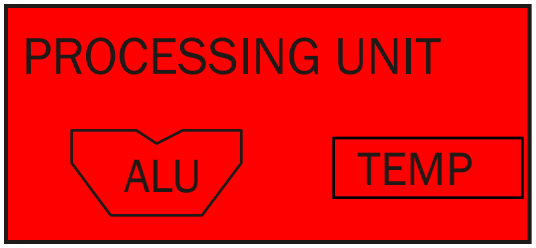
Input and Output on the LC-3

- The LC-3 supports…
  - Input device: Keyboard
    - Data register (KBDR)
    - Status register (KBSR)
  - Output device: Monitor
    - Data register (DDR)
    - Status register (DSR)

What a Program Does

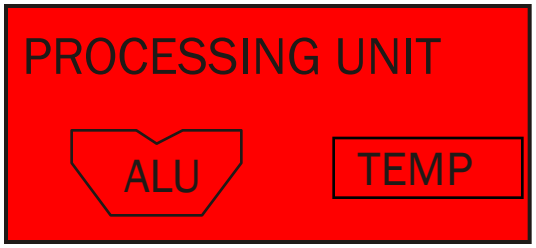
- A program also
  - Evaluates arithmetic & logical functions
  - Determines values to assign to variables
  - Determines the order of execution of the statements in the program
- Two types of instructions
  - Arithmetic / logic
  - Control

A Processing Unit



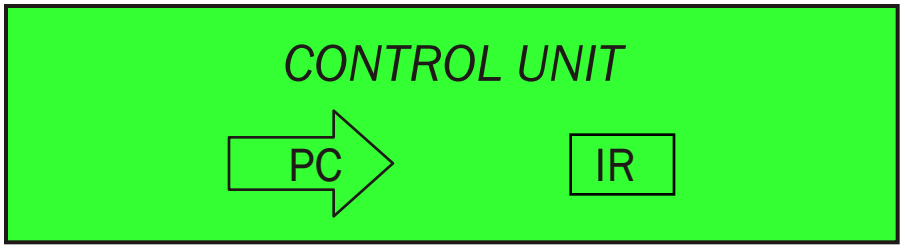
- Functional Units
  - ALU = Arithmetic/Logic Unit
  - A processor could have many functional units, some of them special-purpose
    - E.g., multiply, square root
- Registers
  - Small, temporary storage
  - Operands and results of functional units
- Word Size
  - Number of bits normally processed by ALU in one instruction
  - Width of registers

The LC-3 Processing Unit

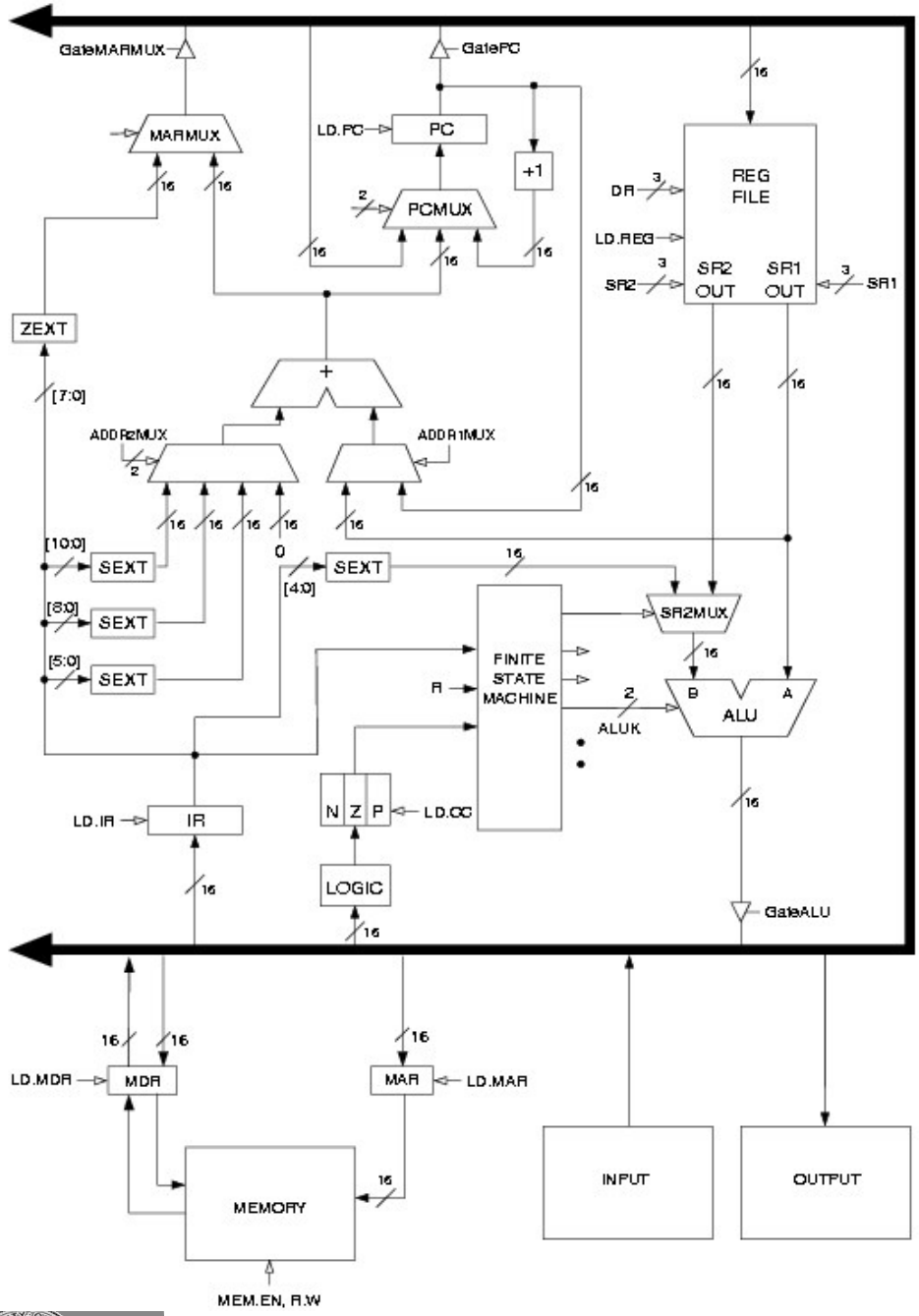


- Functional Units
  - ALU = Arithmetic/Logic Unit
  - LC-3 performs **`ADD, AND, NOT`**
- Registers
  - LC-3 has eight registers (**`R0`**, …, **`R7`**), each 16 bits wide
- Word Size
  - LC-3 has 16-bit words

The LC-3 Control Unit



- Controls the execution of the program
  - Reads an instruction from memory
    - Address in the PC
    - Instruction goes to IR
  - Interpret the instruction
    - Generate signals that tell other components what to do
  - Instructions can take many machine cycles to complete
- Instruction Register (IR) contains the current instruction
- Program Counter (PC) contains the address of the next instruction to be executed



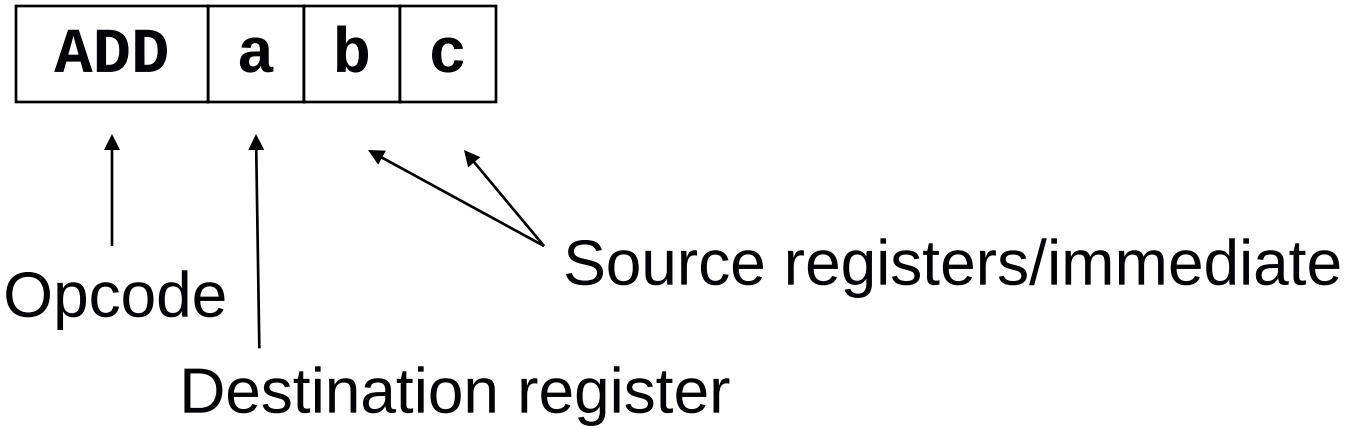
- Can you identify the 5 Von Neumann components?

Instructions

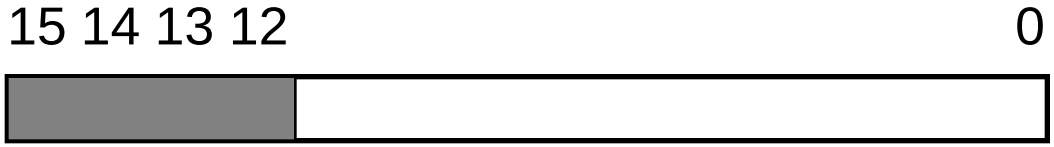
- The instruction is the fundamental unit of work
- Instruction specifies two things
  - Opcode: operation to be performed
  - Operands: data/locations to be used for operation
- Three basic kinds of instructions
  - Computational instructions
  - Data transfer instructions
  - Flow-control instructions

Breaking Down an Instruction

```
ADD  a, b, c
```



Instruction Encoding



- Encoding is the bits' meaning
- In LC-3, the most-significant four bits always contain the instruction's opcode
- The meaning of the other bits changes according to the instruction
- LC-3 has instruction formats (see textbook)

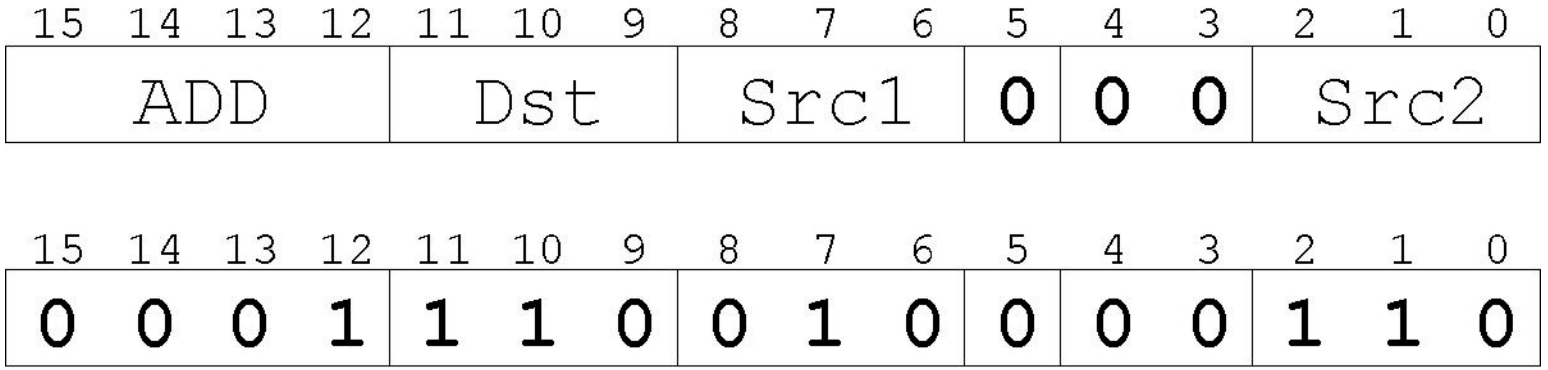
Instruction Encoding

- LC-3 has 8 *registers* (**R0**-**R7**) for temporary storage
  - Sources and destination of **ADD** are registers
- LC-3 has 16-bit instructions
  - Each instruction has a four-bit opcode in bits [15:12]

Computational: LC-3 ADD

LC-3 has 8 *registers* (**R0**-**R7**) for temporary storage

- Sources and destination of **ADD** are registers



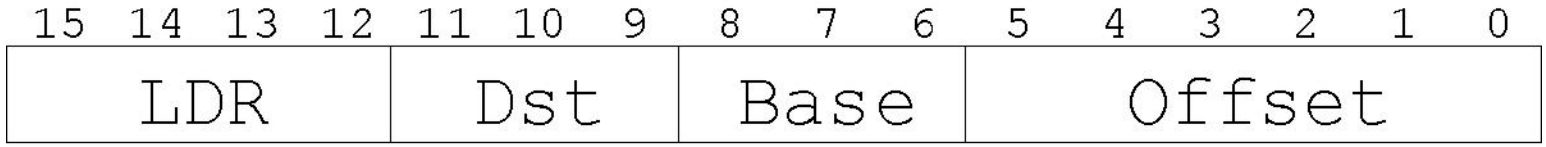
*Add the contents of **R2** to the contents of **R6**, and store the result in **R6**.*

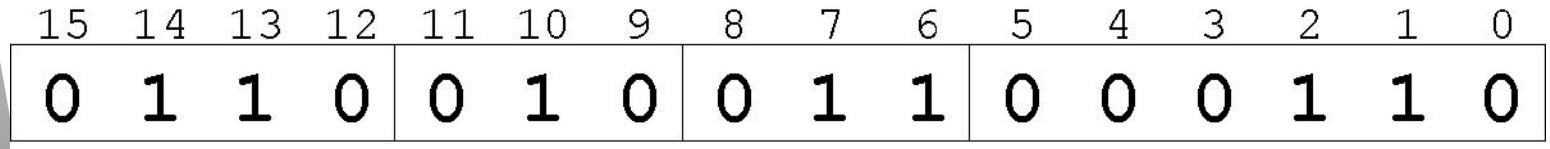
Data Transfer : LC-3 LDR

Load instruction – read data from memory

Base + offset mode:

- add offset to base register - result is memory address
- load from memory address into destination register





Add the value 6 to the contents of R3 to form a memory address. Load the contents of that memory location to R2.

Flow-Control Instructions

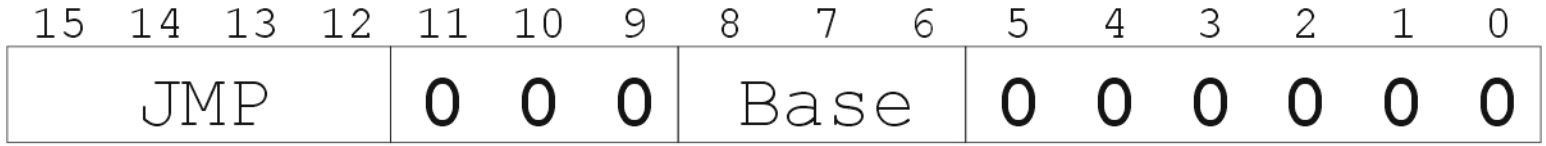
- What if we don't want to always execute the instruction that follows?
  - Loops, if-then statements, function calls
- Question: How do we know what is executed next?
  - Answer: …
  - Need instructions that change the value in …

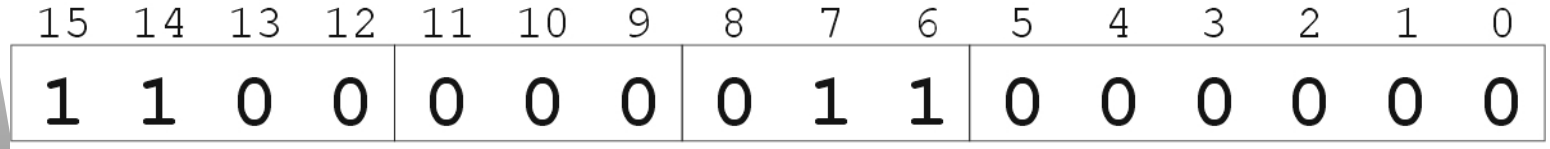
Flow-Control Instructions

- Jumps: unconditional
  - Always change the PC
- Branches: conditional
  - Change the PC only if some condition is true
  - Remember the condition codes register?
    - N
    - Z
    - P

Flow Control: LC-3 JMP

Set the PC to the value contained in a register. This becomes the address of the next instruction to fetch.





*Load the contents of **R3** into the PC.*

Recommended Exercises

- Ex 4.5 (excluding point b3 for now)
- Ex 4.7