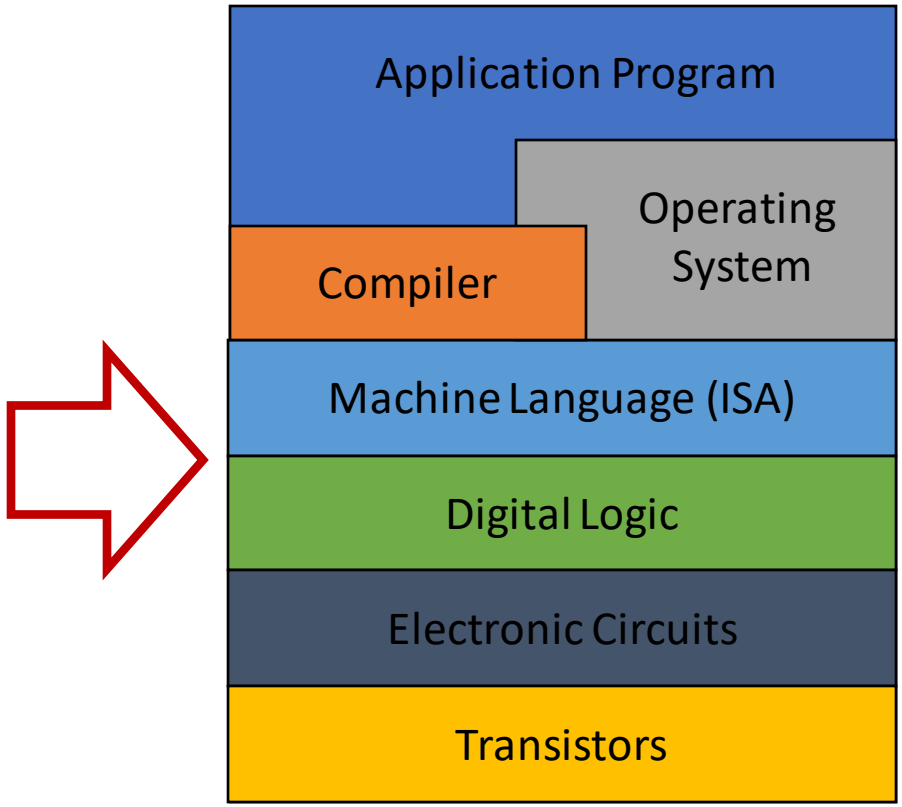
The von Neumann Model

EECS388 Fall 2022

Context

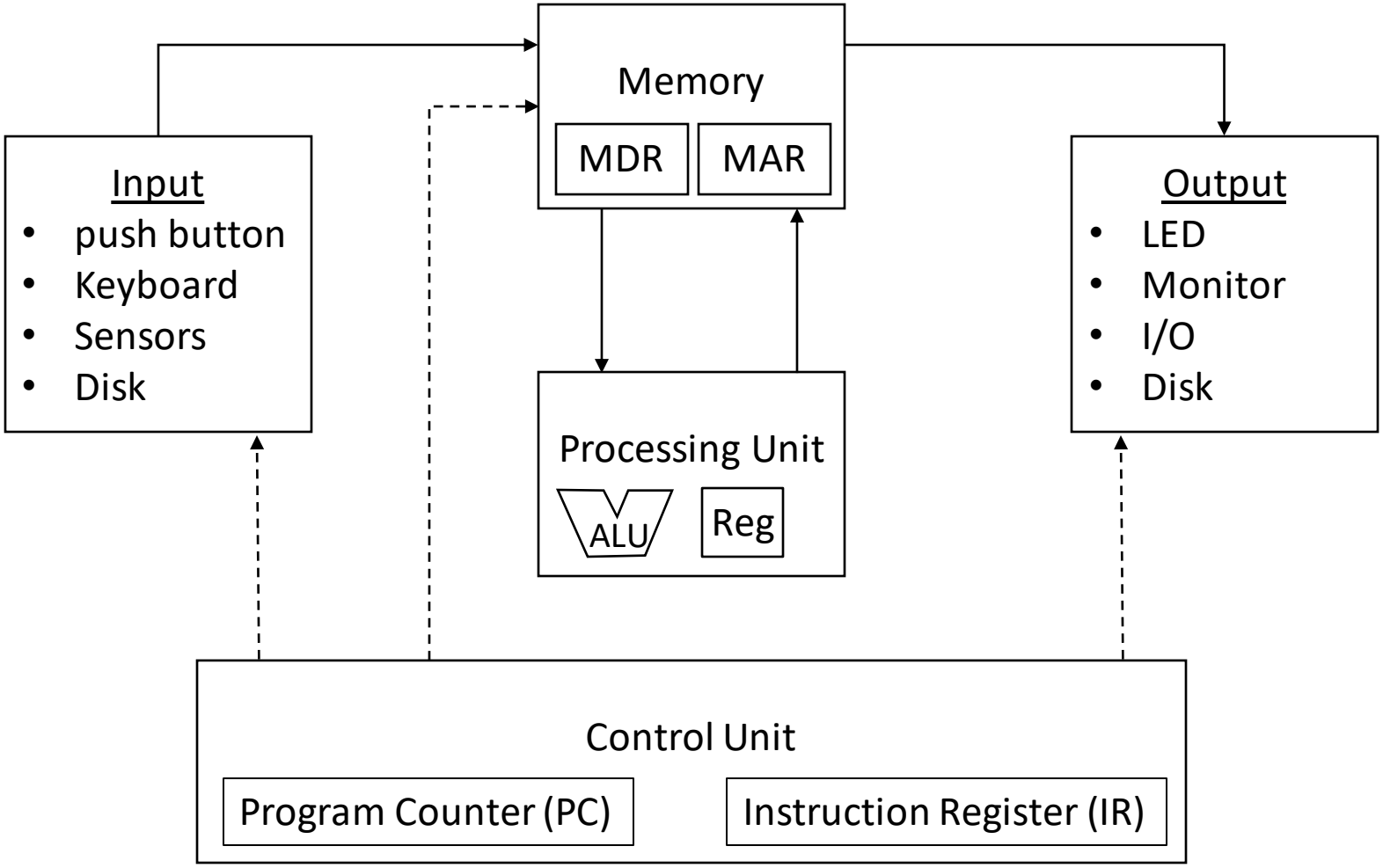
- Recommended reading

Chapter 4 of “Introduction to Computing,” Patt, Patel



For preforming a task by a computer we need:

- A program that specifies the task
  - Instructions: smallest piece of work specified in a computer
- A computer that carry out the task
  - John von Neumann proposed a fundamental model of computer for processing instructions in 1946



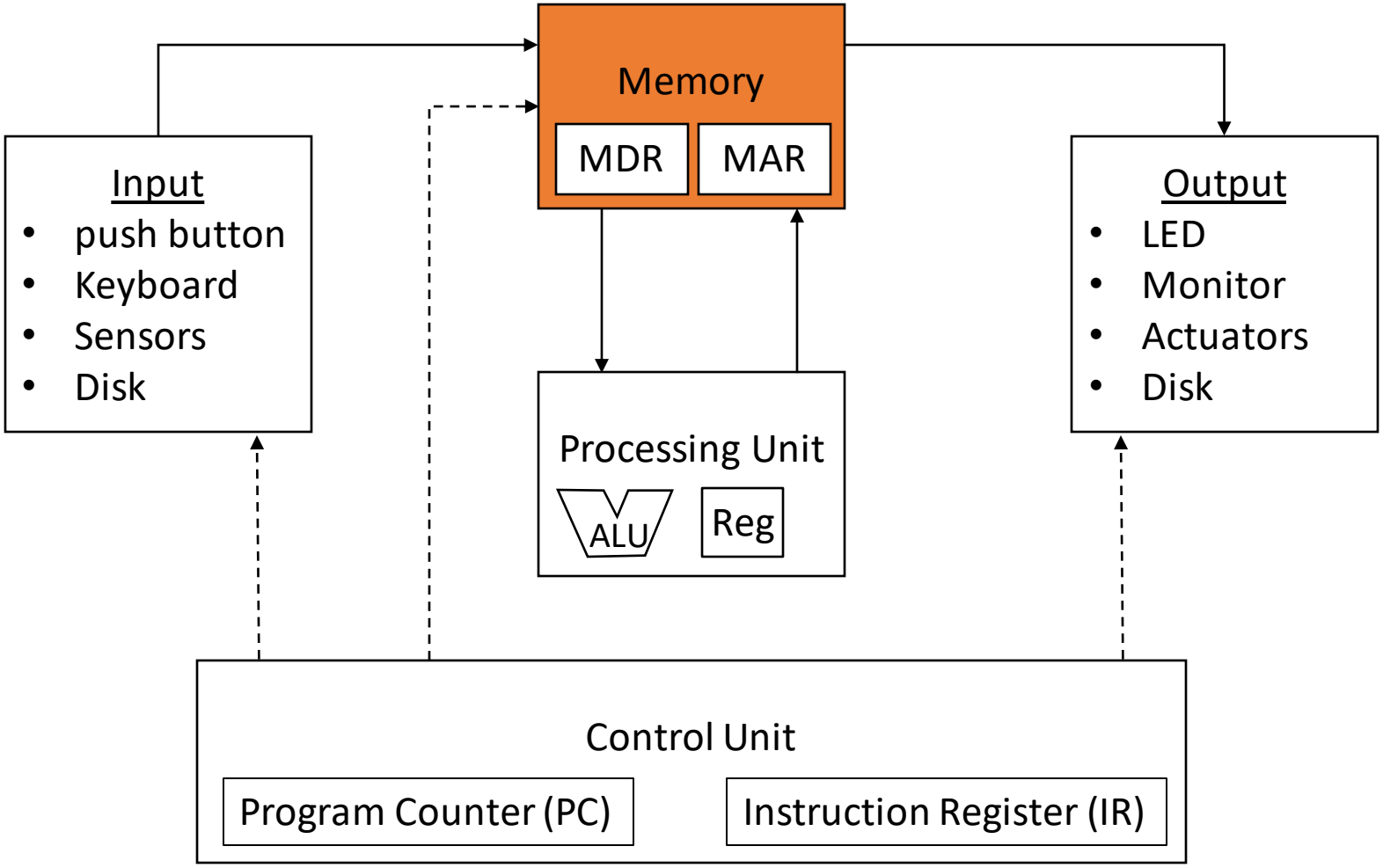
The Von Neumann computer model

Von Neumann model has five parts:

- Memory ← Computer program and data resides here
- Processing Unit ← Processing is carried out here e.g., Add, Sub, compare, …
- Control unit ← Keep track of where we are in the process of running a program
- Input ← Provide the information for processing
- Output ← Store processed data or actuate upon the data

The Little Computer 3 (LC-3)

- We study a simple computer, LC-3
- It has all the important characteristics of the microprocessors
- Just to simplify things …



The Von Neumann computer model

The memory contains bits that stores: Program (instructions) & Data

Each location has

- **Address:** A unique bit patter for identifying a memory location

Address space: Total number of addresses of a memory

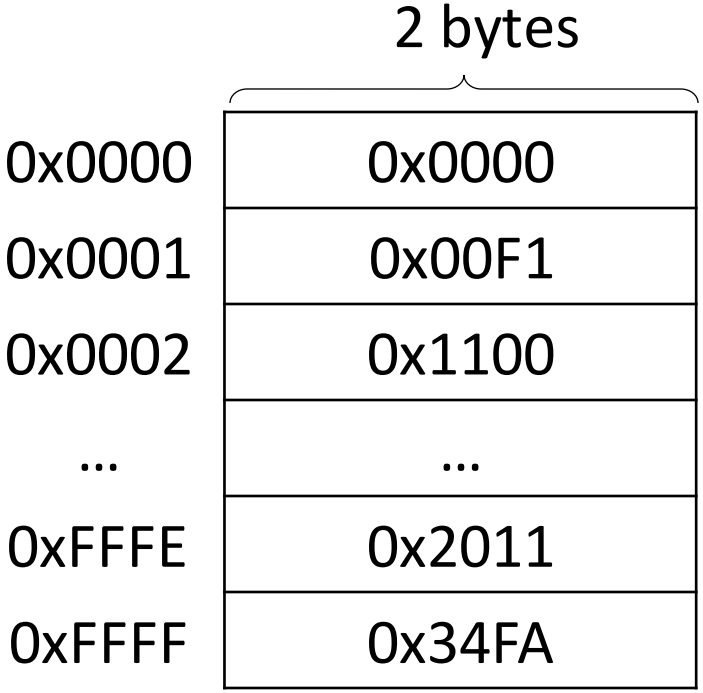
- E.g., n-bit address width => 2^n address space

- **Content:** bits stored at the memory address

Addressability: Number of bits stored at each address

- E.g., 8-bit addressable or “byte addressable”

Example: LC-3 Memory



Address space: of 2^{16} unique memory locations

Addressability: 16 bits or 2 *bytes addressable*

LC-3 Memory Access

- Accessing memory through Memory Address Register (MAR) and Memory Data Register (MDR)

1- Processing unit writes the memory address to MAR

If write

2.1- Processing unit writes the value into MDR and Address in MAR

2.2- Memory will write MDR's content to MAR address

If read

2.1- Memory will read MAR's content to MDR

2.2- MDR is ready to be read by processing unit

How to store multi-byte variables (short, int, long, etc.) in memory?

- int var = 0x49345678;
- assume &var = 0x0;

Big Endian

Address	data	
0x00	0x49	← MSB
0x01	0x34	
0x02	0x56	
0x03	0x78	← LSB
…	…	

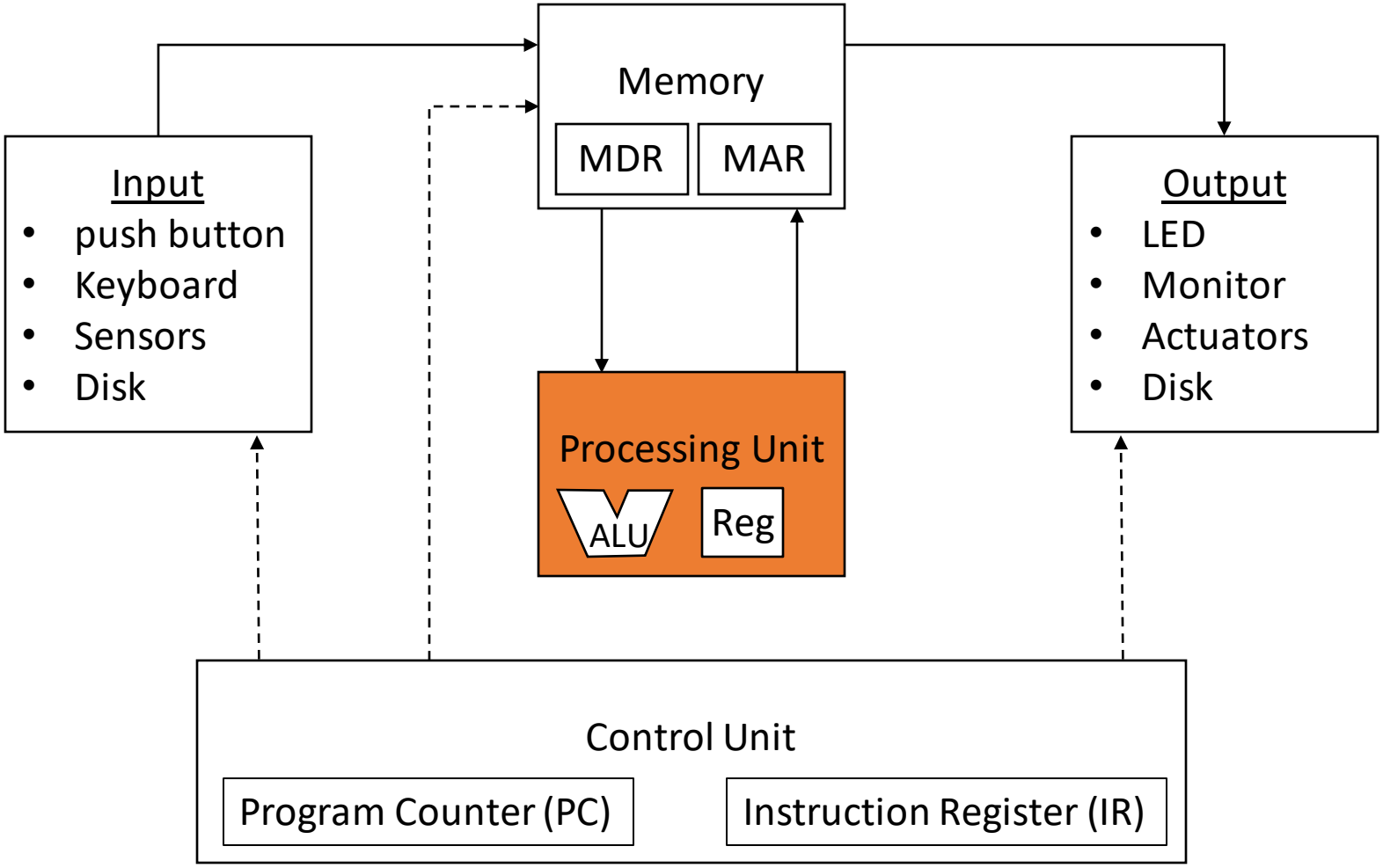
Little Endian

Address	data	
0x00	0x78	← LSB
0x01	0x56	
0x02	0x34	
0x03	0x49	← MSB
…	…	

Little Endian vs. Big Endian

Does it really matter?

Only when two systems with different endianness want to share data



The Von Neumann computer model

Processing Unit

Arithmetic and Logic Unit (ALU)

- Arithmetic functions: ADD, SUB, Multiply, …
- Logic functions: Not, XOR, Shift, Compare, …
- Word length: ALU process data of a fixed size
  - Each ISA has its own word length
  - 64 bit (Intel Core), 32 bit (Intel Atom), 16 bit (LC-3)

Registers

- Small temporary storage close to the ALU
- Operands and results of ALU
- Prevent long latency accesses to memory
- LC-3 has 8 registers
- Number of bits processed in one instruction

Registers (cont.)

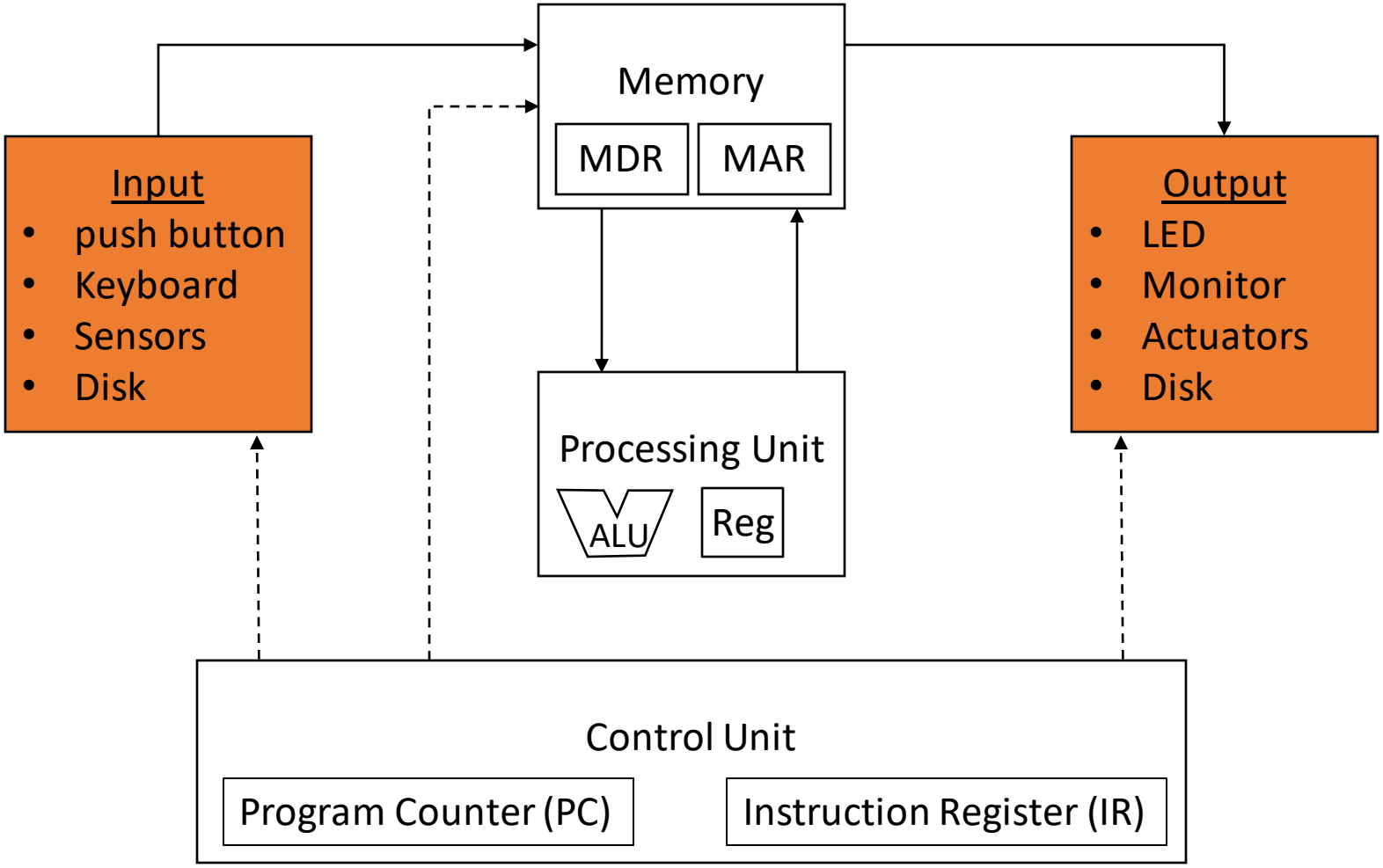
Registers are fast but small

- Memory is large but slow

Register file

- A structure in CPU that holds all the registers
- R0, R1, …, R7 in LC-3

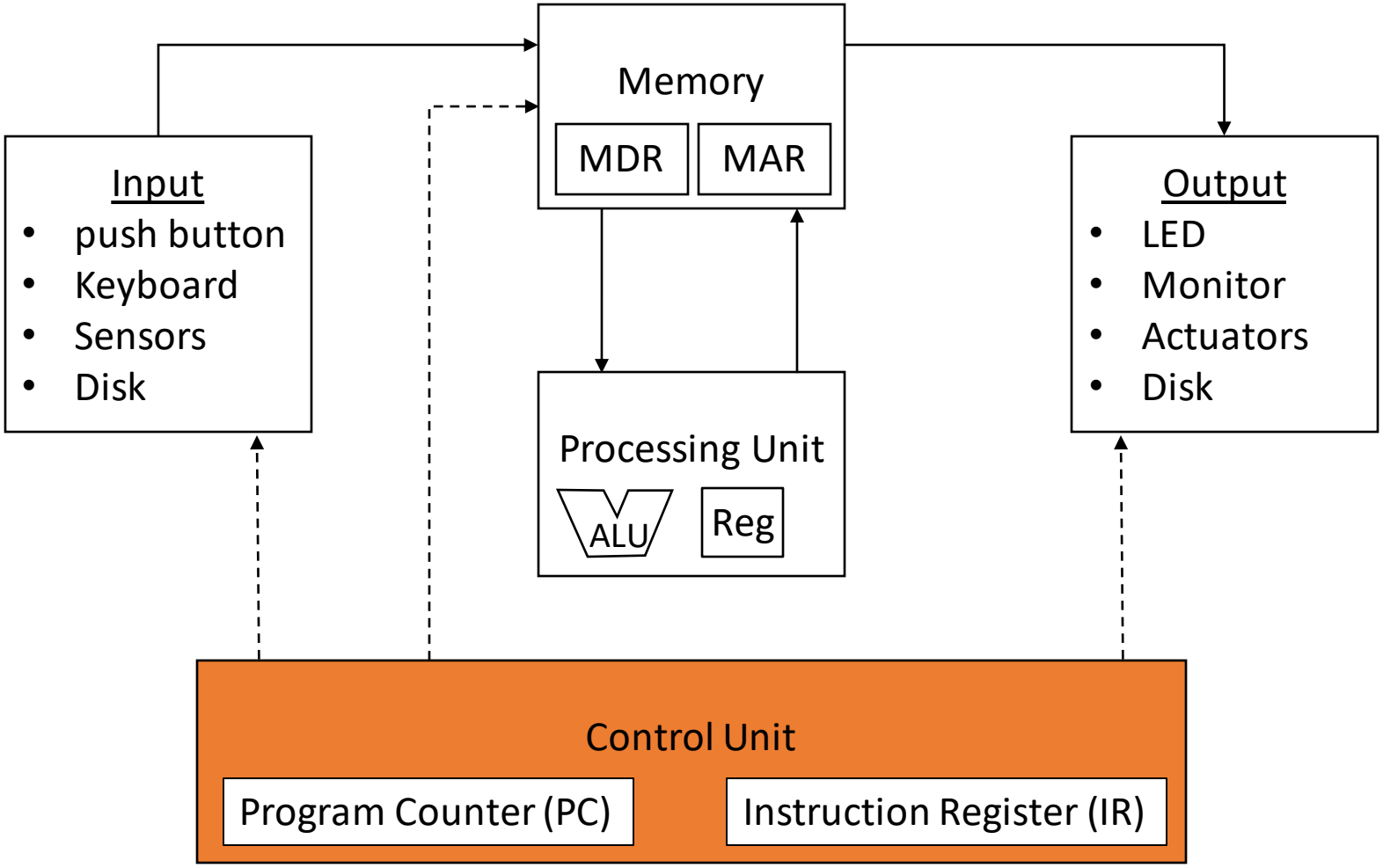
General purpose vs. special purpose register



The Von Neumann computer model

Input and output devices (i.e., peripherals) are used for getting data into and out of computer memory

Usually each device has a set of registers to communicate with the CPU



The Von Neumann computer model

Control unit orchestrates the execution of each instruction

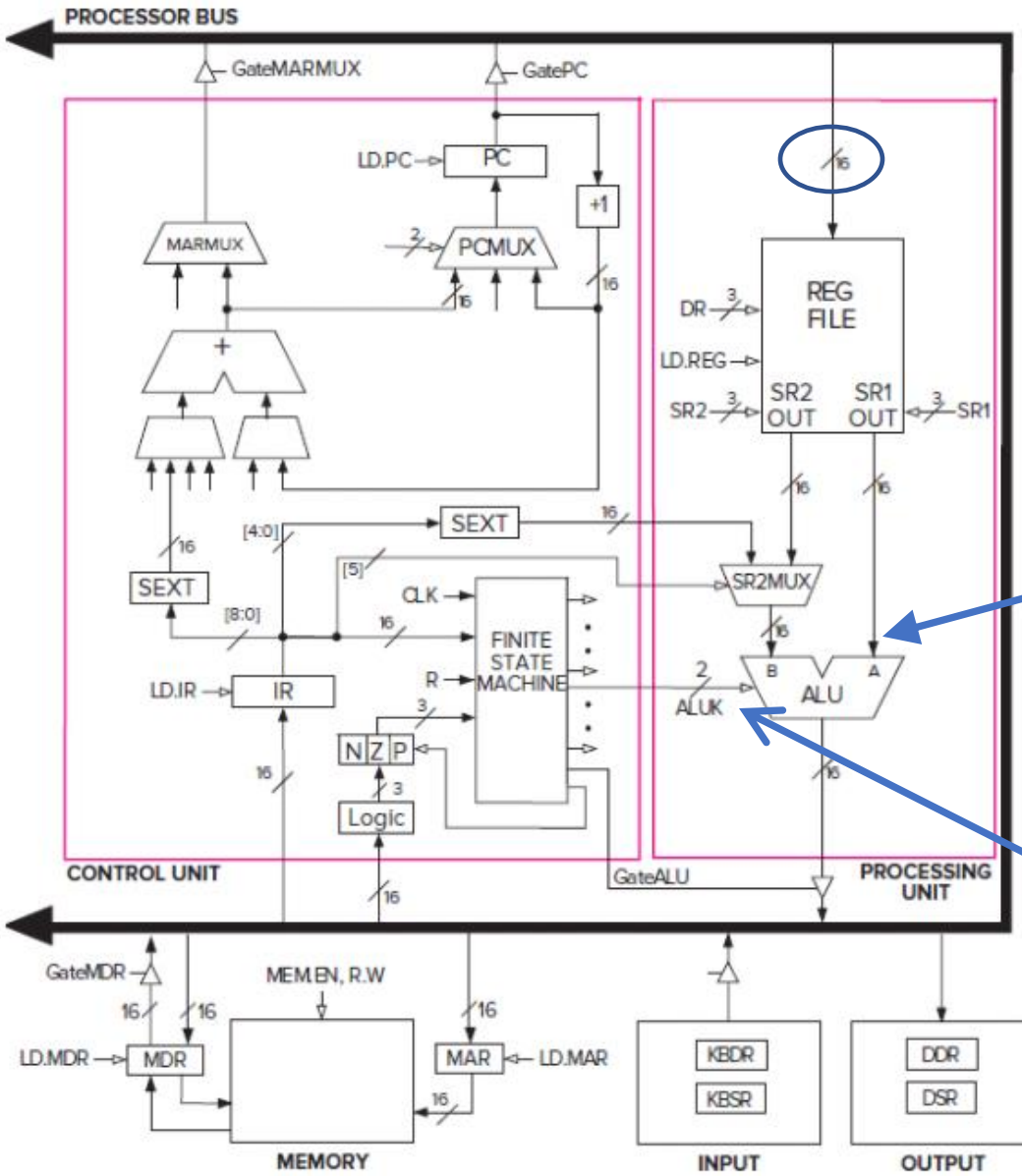
- Instruction Register (IR): the current instruction being executed on the CPU
- Program Counter (PC): the address of the next instruction to be executed

Control Unit at a Glance

- Read an instruction from memory, from the address stored in PC
- Store it into Instruction Register (IR)
- Decodes the instruction
- Generate control signals that tell other components what to do

Picture take from Intro to Computing textbook by Patt Patel.

The LC-3: An Example von Neumann Machine



Signal width

Data signal

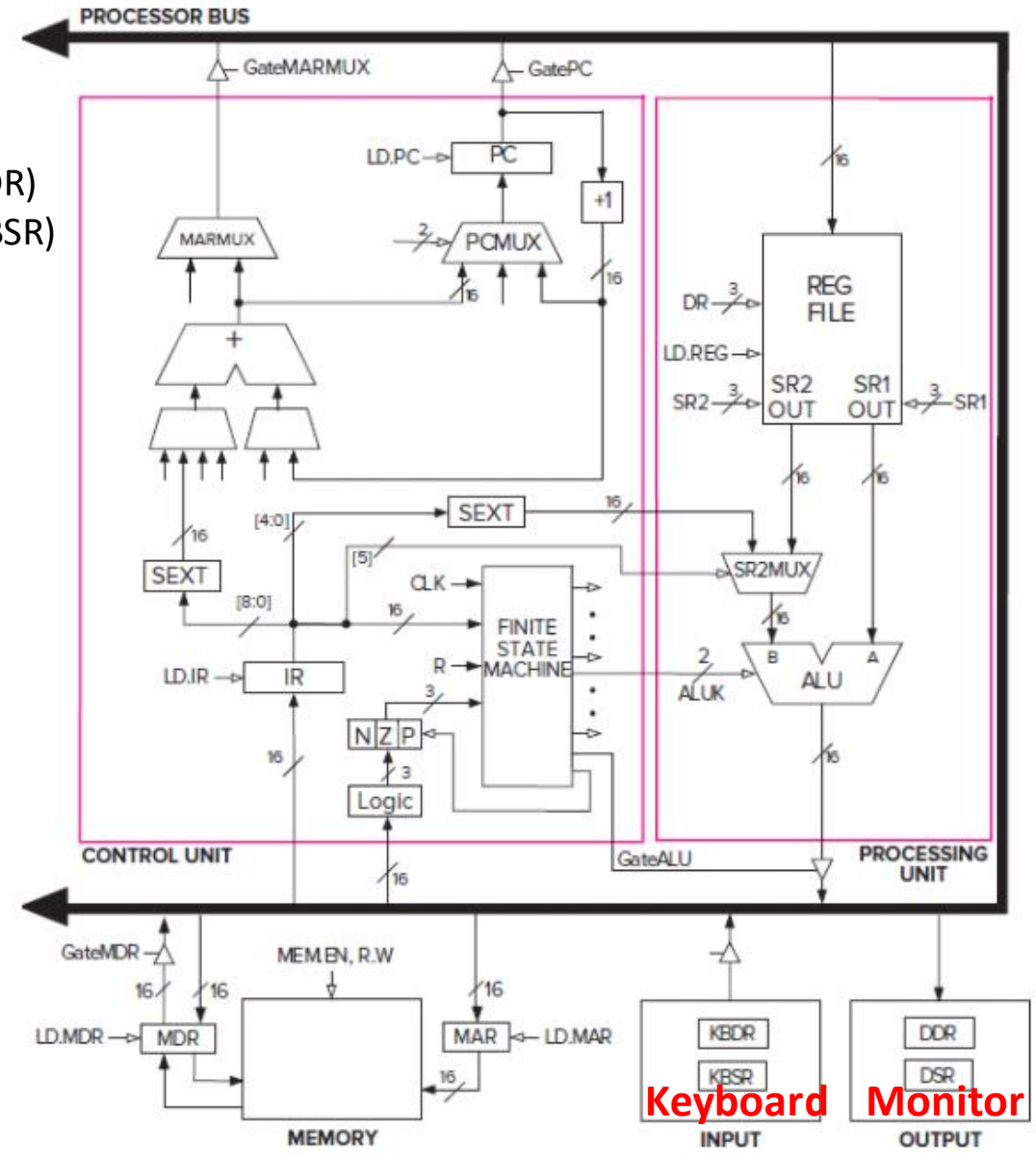
Control signal

Picture take from Intro to Computing textbook by Patt Patel.

The LC-3: Input/Output

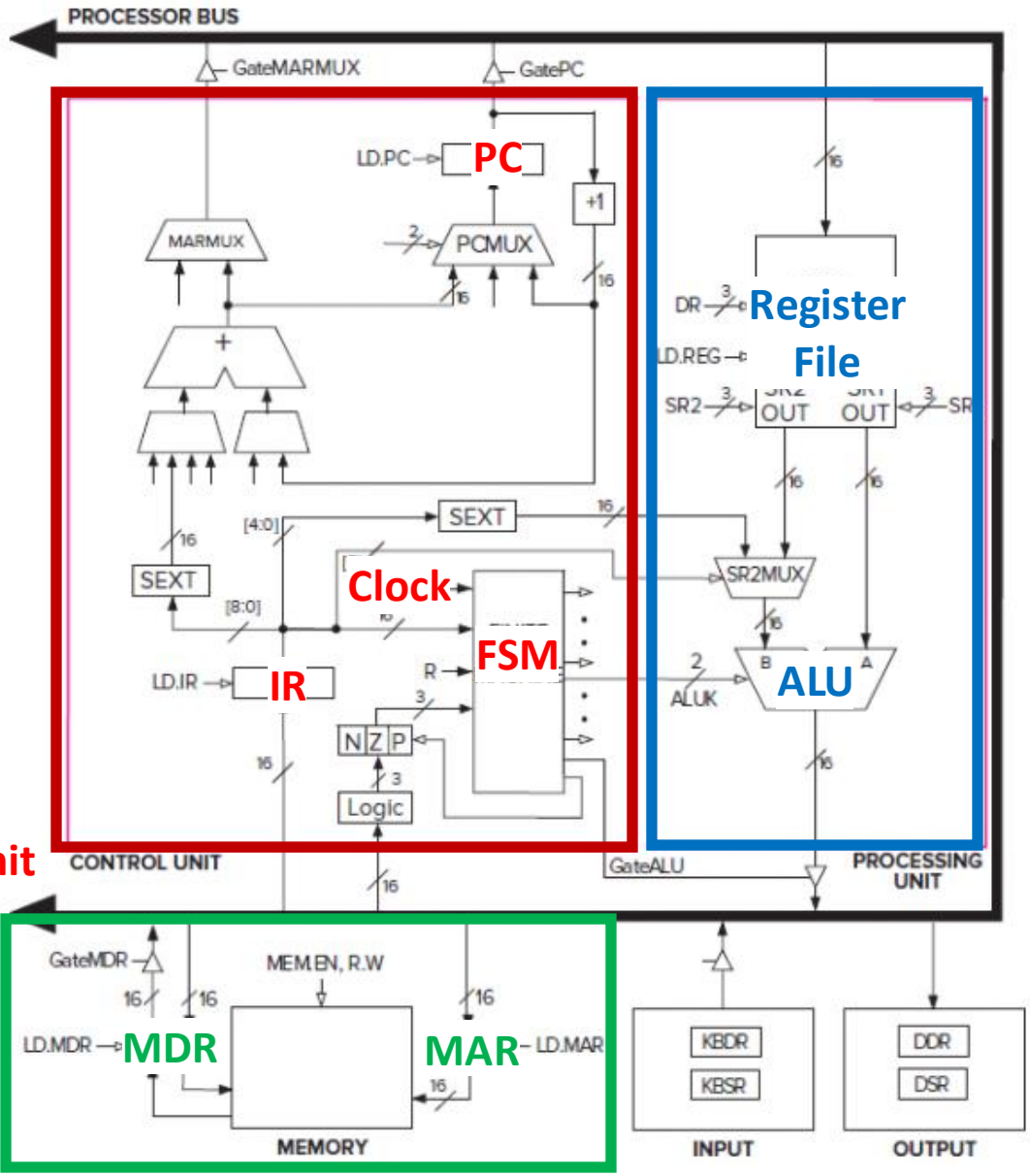
Keyboard
Keyboard Data Reg (KBDR)
Keyboard Status Reg (KBSR)

Monitor
Display Data Reg (DDR)
Display Status Reg (DSR)



Picture take from Intro to Computing textbook by Patt Patel.

The LC-3: Processing Unit, Control Unit, Memory



Instruction

Smallest unit of execution in a processor

- Instructions are sequences of bits that carry two pieces of information
  - Opcode: operation to be performed
  - Operands: data to be used for operation
- In LC-3 we have 15 instructions
  - We need 4 bits for opcode => 12 bits left for operands
    - *Recall the *word length* of LC-3 is 16 bits

Instruction Types

- Operate
  - Operate on data
  - LC-3: arithmetic (ADD), logical (AND, NOT)
- Data movement
  - Move data between processor, memory, and I/O devices
  - LC-3: 6 data movement instructions
- Control
  - Alter the sequential processing of instructions

LC-3 ADD Instruction

- Requires three operands: 2 source (the numbers to be added), 1 destination (the sum)
- At least one of operands from a register and the result stored into a register

🔔Recall: LC-3 has 8 register => 3 bits needed to identify each register

LC-3 ADD Instruction

ADD

ADDi

Source 2 Register

Source 2 Immediate value

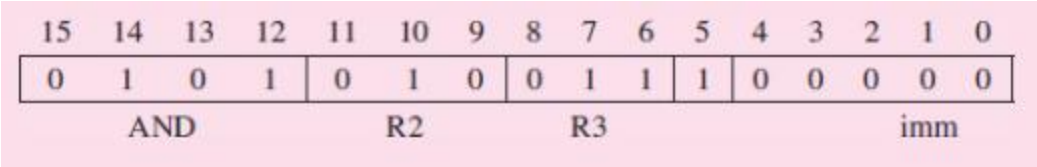
Opcode

Result Register

Source 1 Register

LC-3 AND Instruction

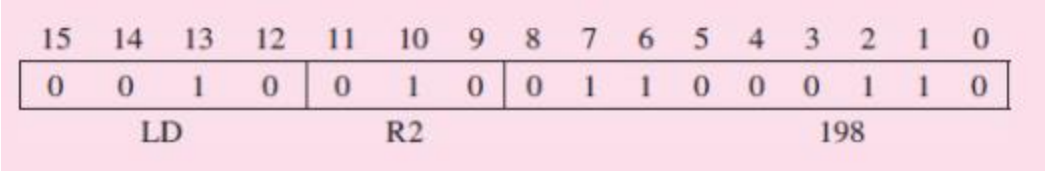
- Opcode: bits [15:12] = 0101 -> bitwise AND



- Sources: R3 and immediate value 0
- Result: Stores 0 to R2
  - Initialization to zero!

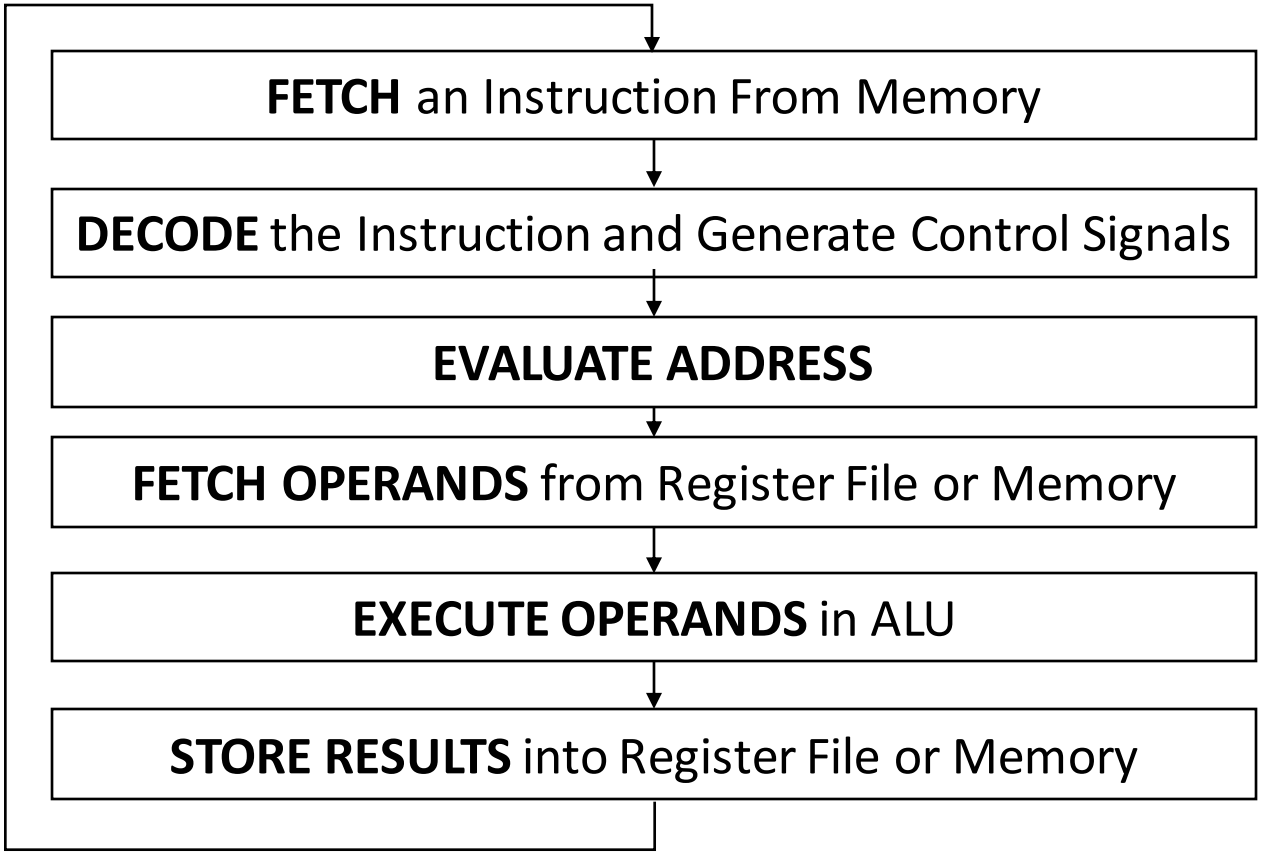
LC-3 Load (LD) Instruction

- Opcode: 0010 -> LD
  - Go to a particular memory address, read the content, store it to the specified register



- Addressing mode: PC + offset
  - Address = sign extended bits [8:0] to 16 bits + PC

Instructions are processed under the direction of control unit in a systematic step by step manner



Instruction Processing Phases

FETCH

- Obtain the next instruction from memory into IR
- PC points to the “next instruction” in the memory

Step 1: Load MAR with PC and Increment PC

Step 2: Interrogate Memory, instruction gets loaded into MDR

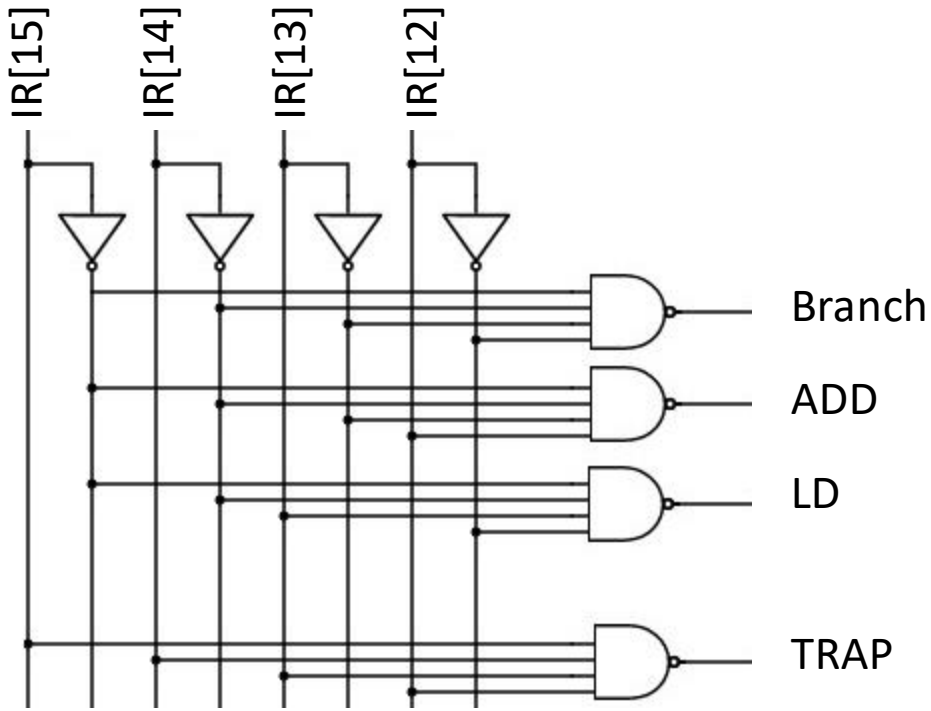
Step 3: Copy MDR to IR

Each step corresponds to one machine cycle

- A 4GHz processor completes 4 billion clock cycles per second!

DECODE

- First determine the instruction type
  - The remaining 12 bits identify what else is needed to process the instruction



EVALUATE ADDRESS

- Only for memory access instructions
- E.g., for LD, sign extend bits [8:0] and add it to PC

FETCH OPERANDS

- Obtain the source operands needed to process the instruction
- E.g., for LD takes two steps: load address (calculated in “Evaluate Address” phase) to MAR, then interrogate memory
- E.g., for ADD, reading from reg file

EXECUTE

- Executing the instruction
- E.g., ADDing two operands in ALU

STORE RESULTS

- The instruction result is written to its destination
- E.g., for ADD, update the register file

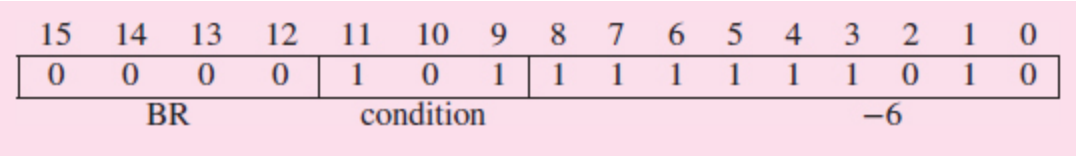
- Optimization: perform “Fetch Operand”, “Execute” and “Store Result” phases in one cycle for ADD!

Changing the Sequence of Execution

- We talked about ***operate*** and ***data movement*** instructions so far
- ***Control instructions:*** change the sequence of instruction execution
  - Alter the value of PC in EXECUTE phase
- *E.g., Branch (BR)*

LC-3 Conditional Branch Instruction

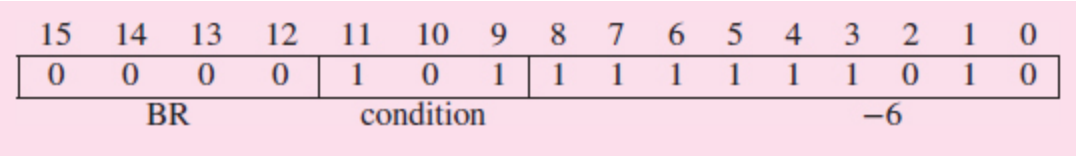
- Opcode: 0000
  - Condition: 101 -> (none zero) taken branch if previous instruction's results is not zero




```
R2 <- 0
R1 <- 1
R3 <- R2 + R1
//Above BR behavior?
```

LC-3 Conditional Branch Instruction

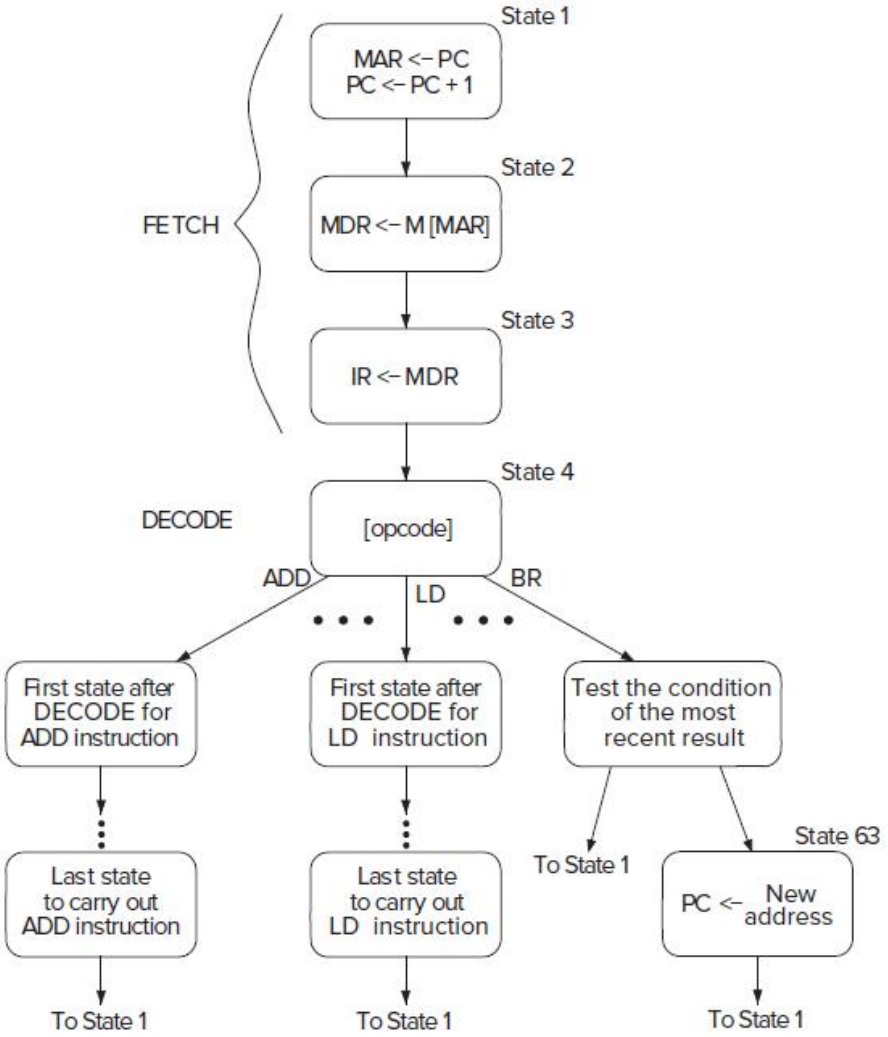
- Opcode: 0000
  - Condition: 101 -> (none zero) taken branch if previous instruction's results is not zero




```
R2 <- 0
R1 <- 1
R3 <- R2 + R1
//Above BR behavior? Taken to "PC – 6"
```

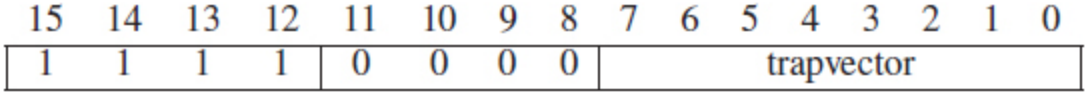
Picture take from Intro to Computing textbook by Patt Patel.

Control of the Instruction Cycle



Halting the Computer

- What if we want to stop the processing cycle?
- TRAP instruction
  - Hand over the execution to the Operating System by explaining the reason using an 8 bit *trap vector*



Recap

- Von Neumann computer model
  - Processing Unit, Memory, Control Unit, I/O
- LC-3
- Instruction
  - Operate, data movement, control
- Instruction processing phases
  - Fetch, Decode, Eval Addr, Fetch Operands, Execute, Store Result
- Control of instruction cycle