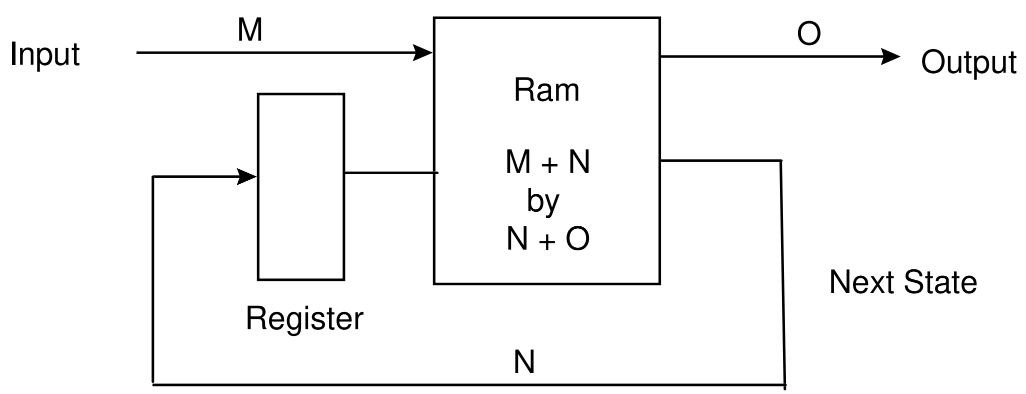
A Note about HW #5

- Remember that the mantissa can be normalized
  - remove leading zeros
- Normalized numbers all have a 1 for the first digit of the mantissa
  - since the one is present in all numbers we do not need to include it
  - the leading one is implicit
  - use an implicit one in your answer to HW#5, question1

Representing a "Program"

- How do we build a circuit that can run different programs?
  - Could write "programs" as state machines
    - Machine could be wired with all possible state machines
    - select one to execute
  - Could build a machine to execute FSMs
    - store the FSM in memory
    - use a register to record the current state



Limitations of FSM “Computer”

- There are limits on what can be computed using just FSM (regular expressions)
- Computers don’t have to accept inputs on every instruction
- How do we load the “program”?
- State machines are not a very natural way to program.

Stored Program Computer

- a sequence of micro-operations as the basic operations
- define a set of standardized operations called instructions
  - each instruction is a sequence of micro-operations
  - instructions are sometimes called macro-operations
- what instruction to execute next
  - normally it is the next instruction
  - provide control to select another instruction as the next instruction

What do Instructions Look Like?

- In this class, instructions:
  - are 16 bits long
  - contain two parts:
    - 4 bit Opcode (indicates what to do)
      - implies there are at most 16 operations
    - 12 bit Address (indicates where to do it)
      - at most 4096 words of memory
      - if the Opcode does not refer to memory can use these bits to encode additional instructions
  - can operate on:
    - an accumulator (a register)
    - memory
    - I/O

Types of Memory References

- Direct memory reference
  - address field in the instruction indicates the memory location to use
- Indirect memory references
  - address field of the instruction contains the address of the memory location that contains the location to use
  - sort of like pointers in C or Pascal
- Use one bit of the op-code to indicate memory addressing mode
  - 0 implies direct reference
  - 1 implies an indirect memory reference

How will we build a machine to run our instructions?

- Use a collection of registers to hold:

  AC Accumulator

  DR Data Register (holds memory operands)

  AR Address register (12 bits)

  IR Instruction Register

  PC Program Counter -address of instruction (12 bits)

  TR Temporary register

  INPR Input Register (8 bits)

  OUTR Output Register (8 bits)

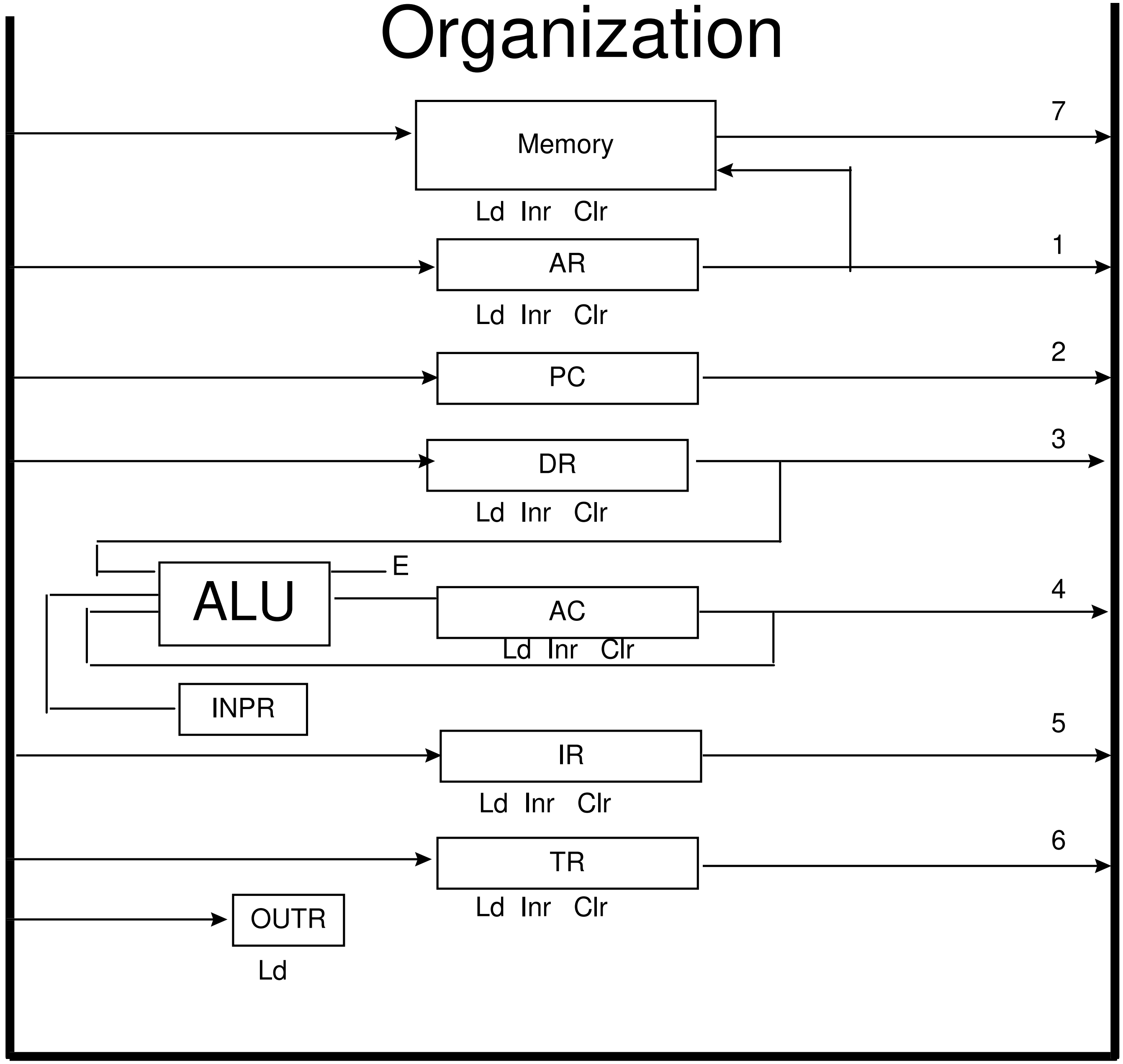
- Have Memory to store:
  - program
  - data

Computer Organization

In addition to registers and memory:

- Have a bus to move information around
  - 16 bits wide
- ALU to compute results
  - also uses 16 bit operands
- Controller to make it all work

Organization



Examples of micro-operations

- AR <- IR
  - S = 101, LD_{AR}=1
- PC <- PC + 1
  - INR_{PC}=1
- DR <- AC, AC <- DR
  - S=100, LD_{DR} =1, ALU (F=A), LD_{AC}=1

Ways to Attach ALU

- ALU needs two inputs
- Could put two registers at ALU
  - bus is free during ALU operations
- One Register at ALU
  - second operand from the bus
- Two buses (one for each operand)
  - permits any register to be the input to the ALU