INF-1100
The Von Neumann model

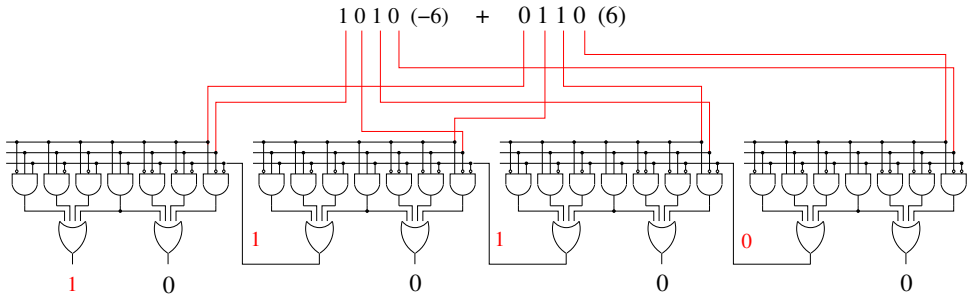
Åge Kvalnes

University of Tromsø, Norway

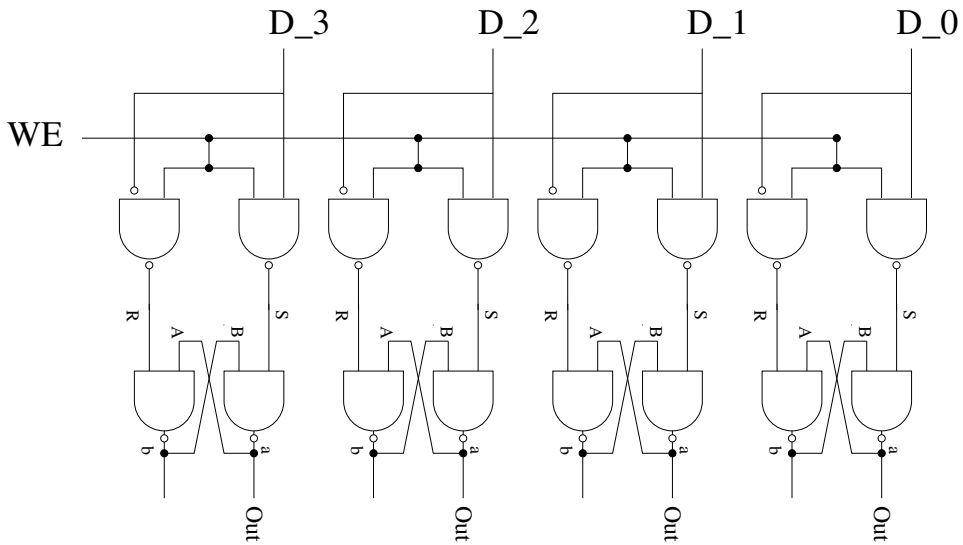
September 11, 2014

What have we learned so far?

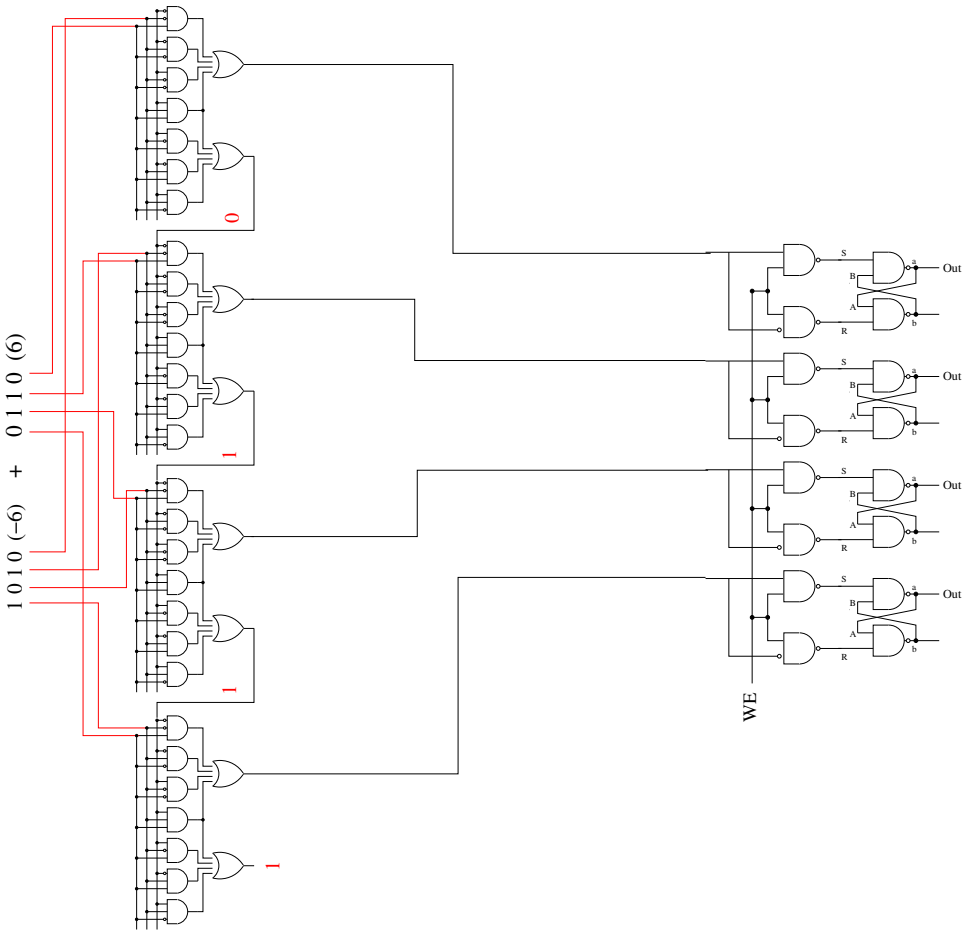
We can build combinatorial circuits that perform addition:



We can build sequential circuits that store data:



Combining



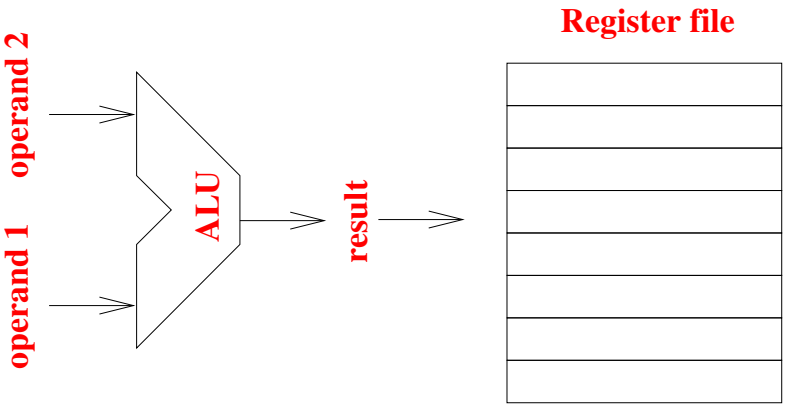
Concept: Compute with combinatorial circuit and store result in sequential circuit

Generalizing: ALU and register file

ALU (Arithmetic logic unit)

Combinatorial circuits:

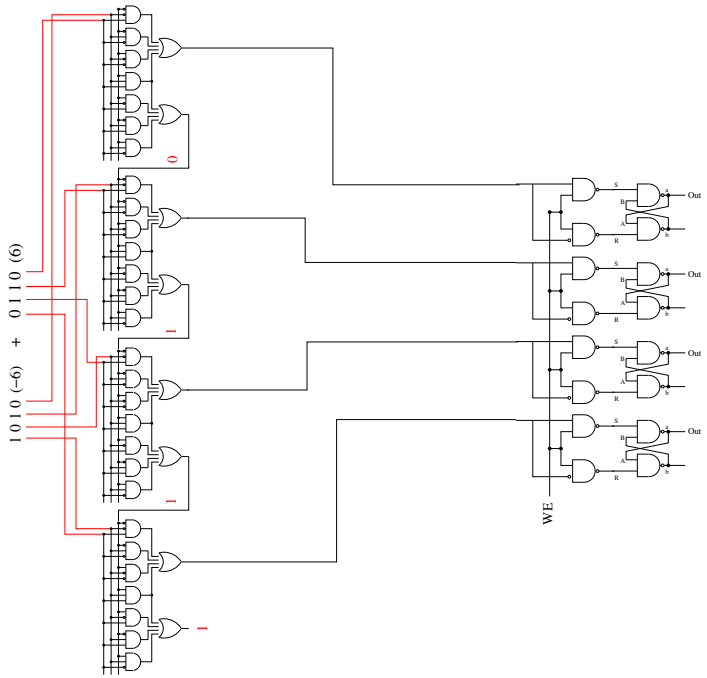
- Two operands as input
- addition, subtraction, etc.
- OR, AND, etc.



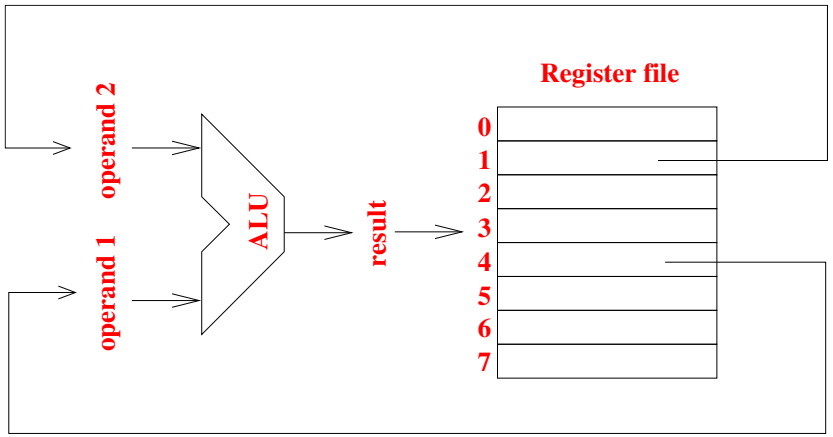
Register file

Sequential circuits:

- Stores ALU output
- Typically 8-128 registers
- Can provide operands to ALU



Generalizing: ALU and register file



Input to the ALU can come from registers

We can imagine a sequence of operations (random example):

- Add contents of register 4 with 0, and store result in register 2.
- Subtract contents of register 2 from register 5, store result in register 6.
- Perform logical OR between contents of register 6 and 3, store result in register 6.

How do we specify what to do? $\Rightarrow$ **Instructions**

The instruction

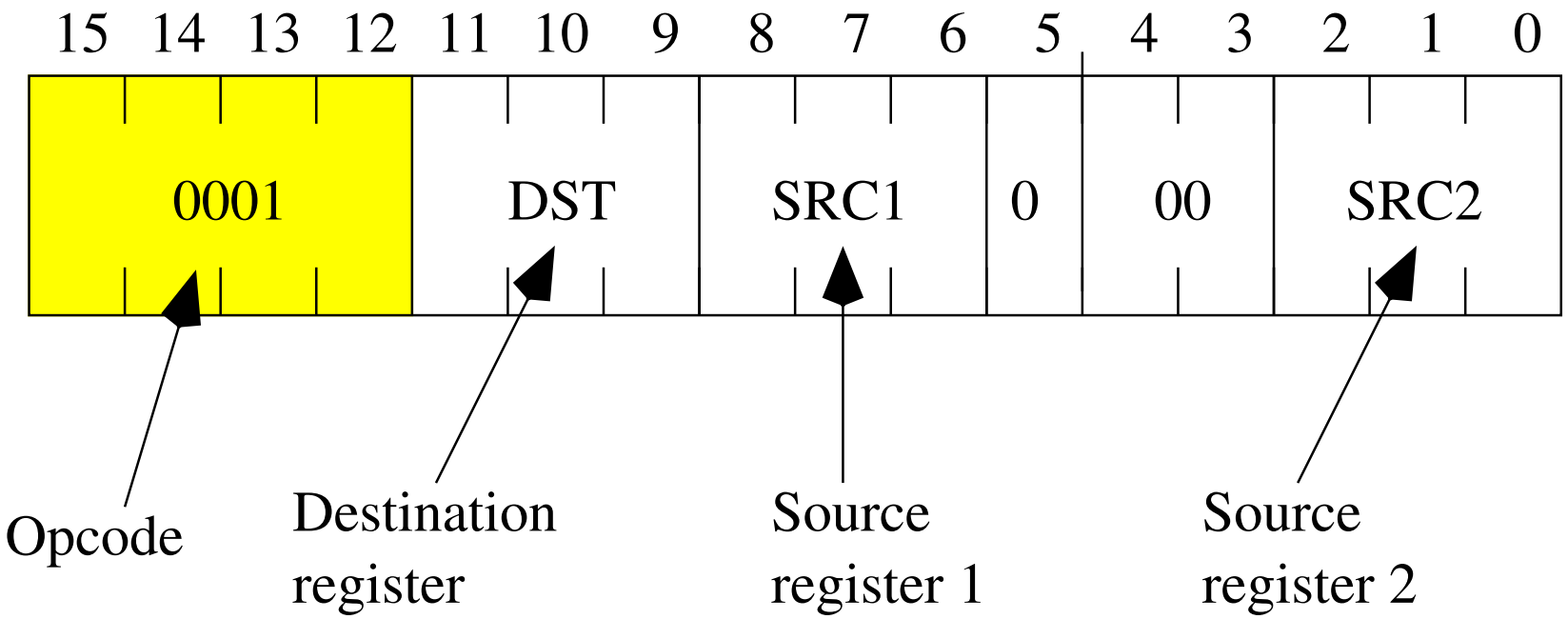
An instruction specifies:

- OPCODE: The operation to perform.
- OPERANDS: Where to find operands for the operation.

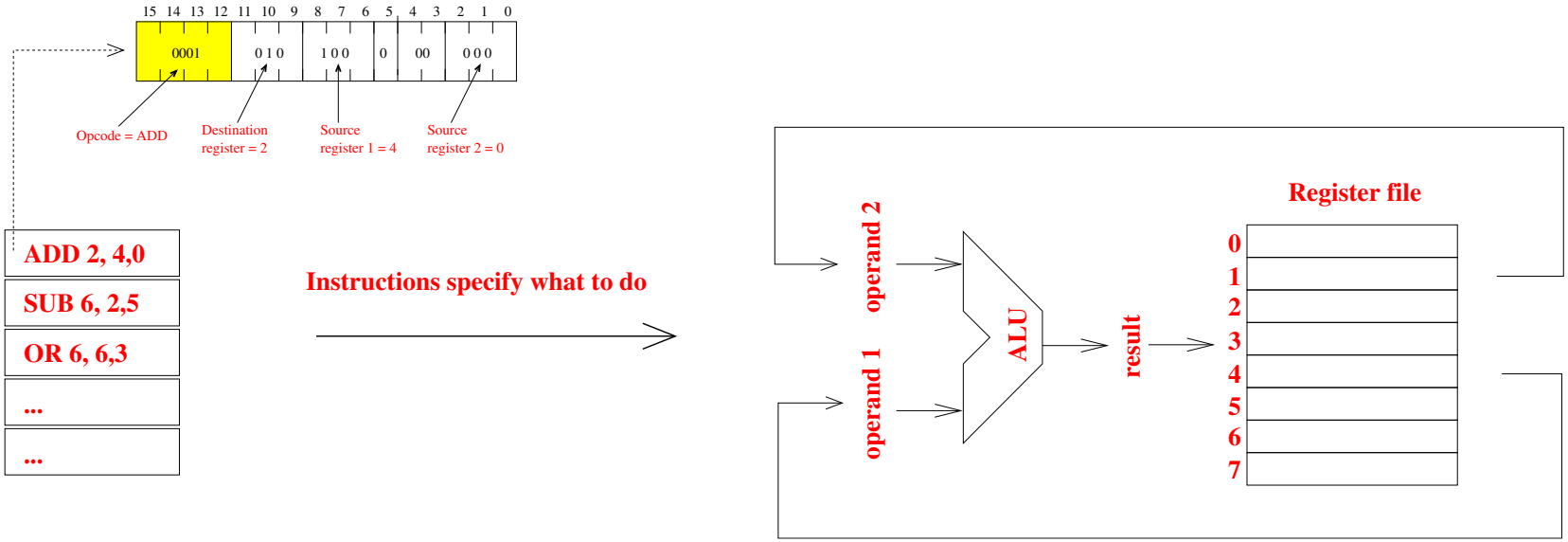
Instructions are encoded as sequences of bits.

- Instructions are typically encoded using a fixed number of bits, e.g. 16 or 32.

Example LC-3 instruction: ADD

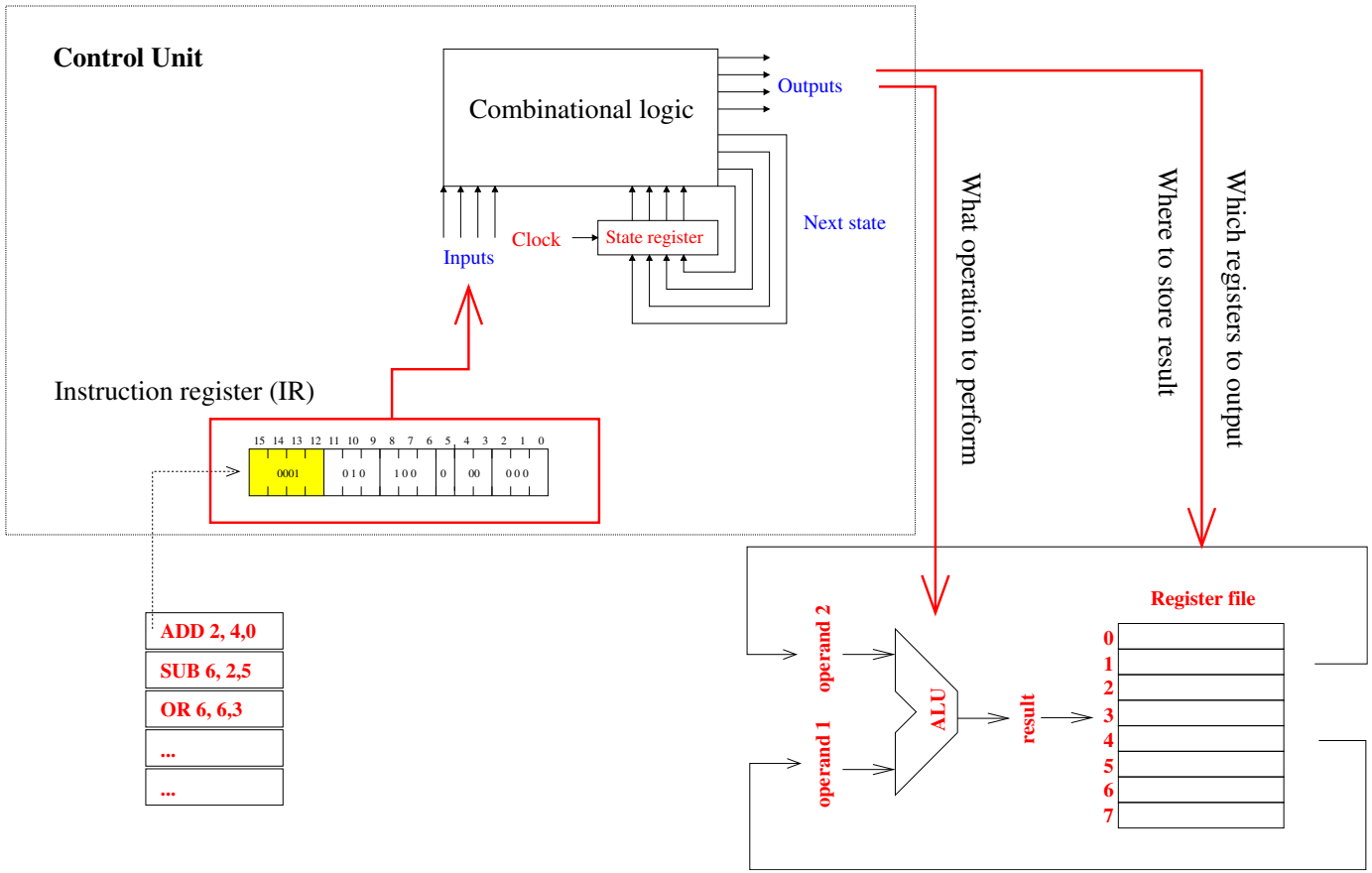


Instructions specify what to do and in what order



How do we control instruction execution? $\Rightarrow$ **The control unit**

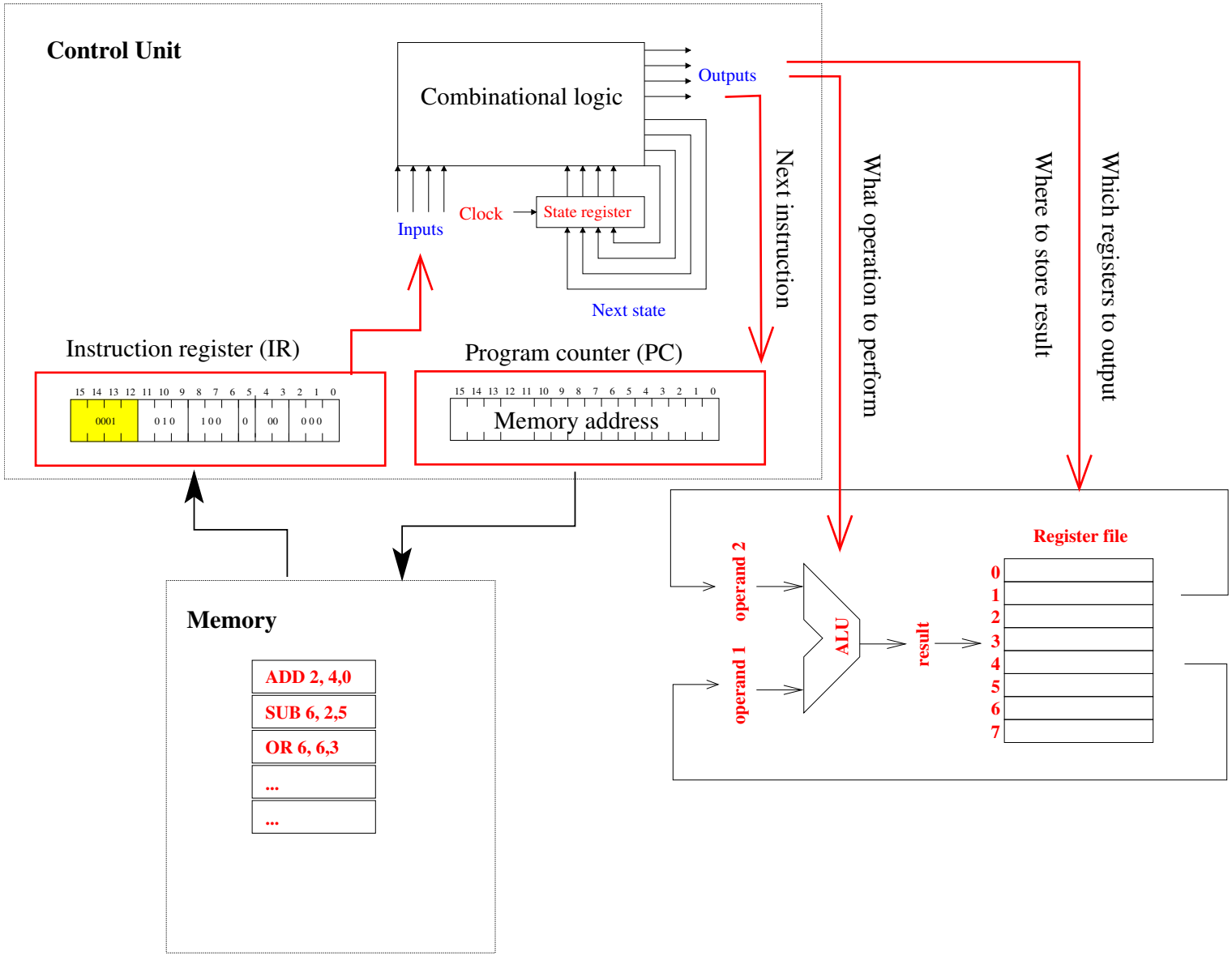
The Control Unit



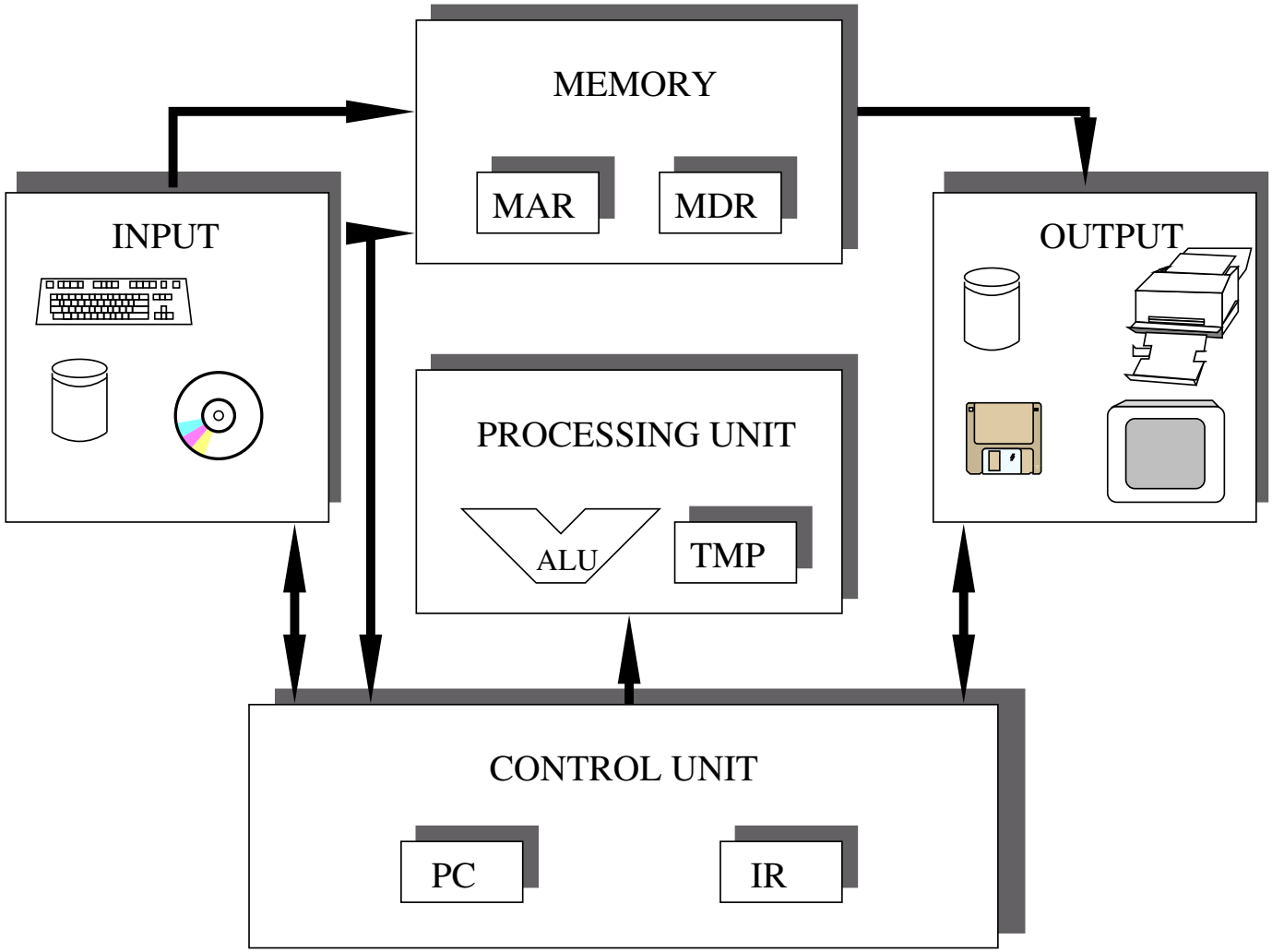
The control unit is a state machine:

- INPUT: Sequence of bits describing instruction to execute
- OUTPUT: What ALU operation to perform, which registers to send to ALU

Instructions are stored in memory



The big picture: The Von Neumann model



Memory

Memory is essentially a $k * m$ matrix of bits.

Address space:

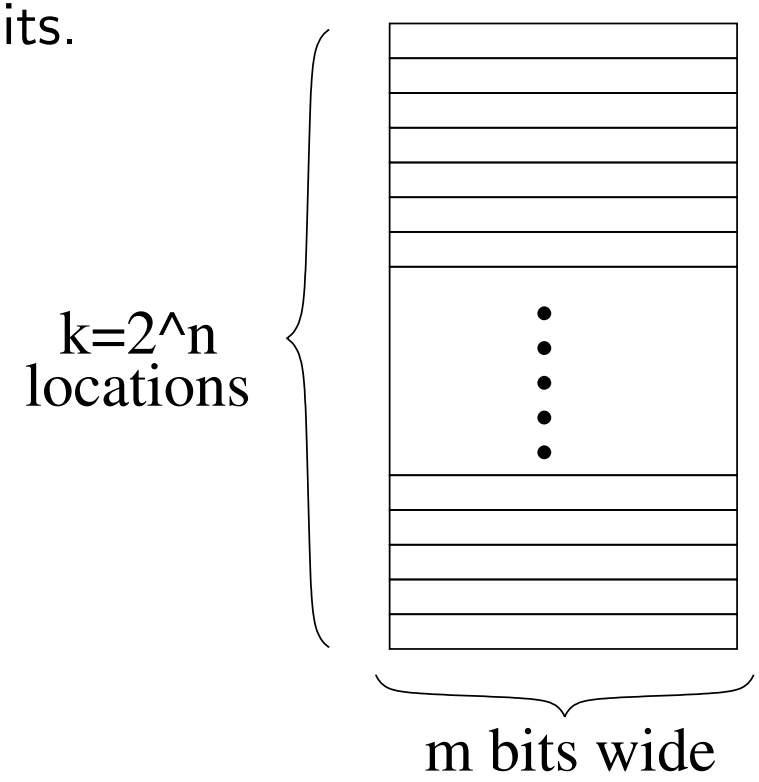
- Number of locations (k).
- Usually a power of 2.

Addressability:

- Number of bits per location.
- Usually 8 bit (a byte).

Operations:

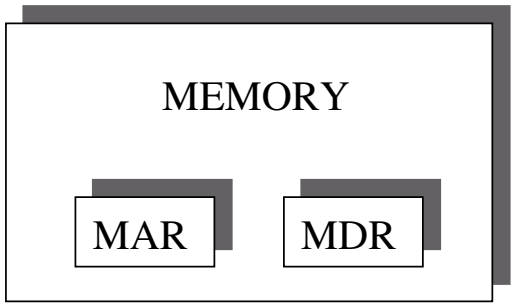
- READ: Read contents of a memory location.
- WRITE: Write a value to a memory location.



Interfacing with memory

MAR: Memory address register.

MDR: Memory data register.



Reading from memory:

- Place address in MAR.
- Send READ control signal to memory.
- Read memory content from MDR.

Writing to memory:

- Place address in MAR.
- Place value in MDR.
- Send WRITE control signal to memory.

Processing unit

ALU: Arithmetic Logic Unit.

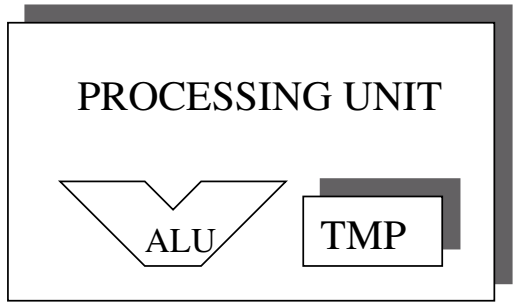
TMP = registers.

ALU:

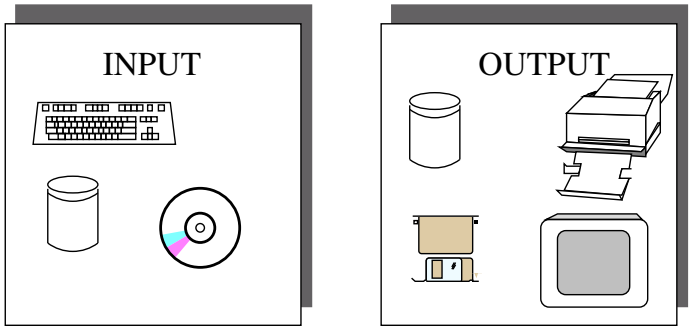
- Logic for performing arithmetic and logical operations (ADD, SUB, OR, etc.).

Registers:

- Small and fast memory (SRAM).
- Used for intermediate results.



Input and output



Devices for moving data to and from memory.

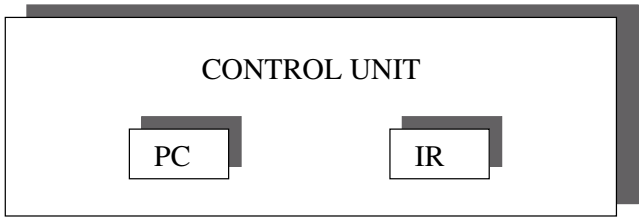
Each device contains a set of registers that are used to both control and move data to and from the device (similarly to the MDR and MDA memory registers).

- Example: LC-3 Keyboard (input)
  - KeyBoard Data Register (KBDR)
  - KeyBoard Status Register (KBSR)
- Example: LC-3 Console (output)
  - Display Data Register (DDR)
  - Display Status Register (DSR)

Control unit

IR: Instruction Register.

PC: Program Counter.



The control unit orchestrates the execution of instructions.

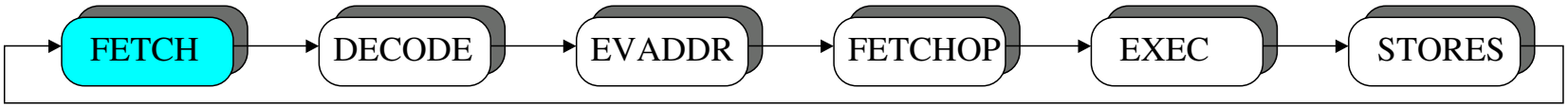
IR:

- Holds the current instruction.

PC:

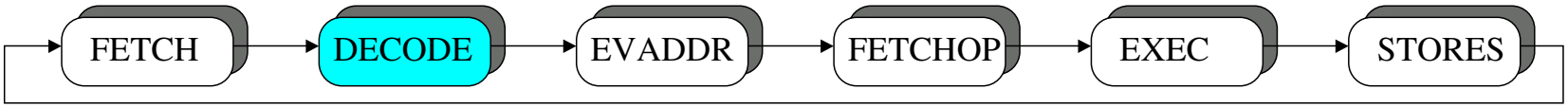
- Holds the address of the next instruction to execute.

Instruction cycle: FETCH

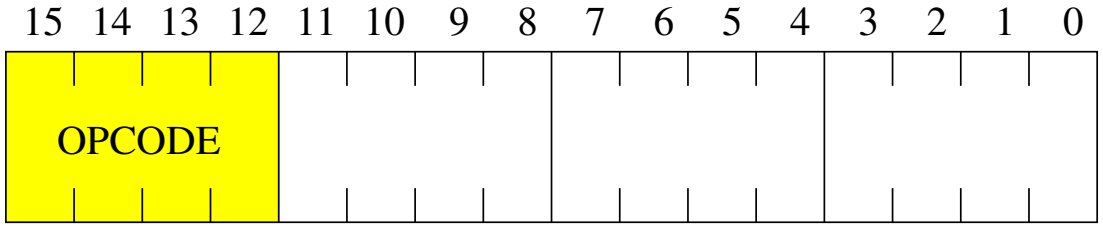


- Fetch next instruction from memory and place it in instruction register (IR).
  - Place address of instruction in memory address register (MAR).
  - Memory responds by placing data in memory data register (MDR).
  - Contents of MDR copied to IR.
- Increment program counter (PC) to point to next instruction.

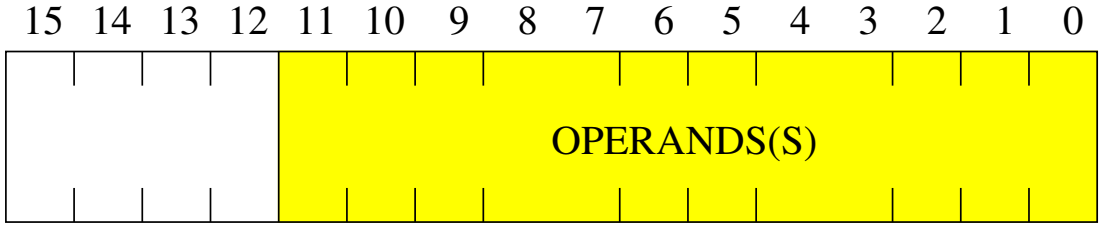
Instruction cycle: DECODE



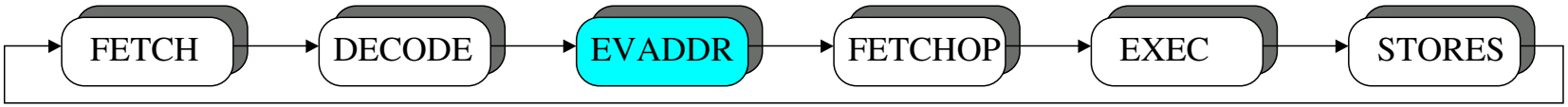
1. Determine opcode



2. Based on opcode, determine operands.



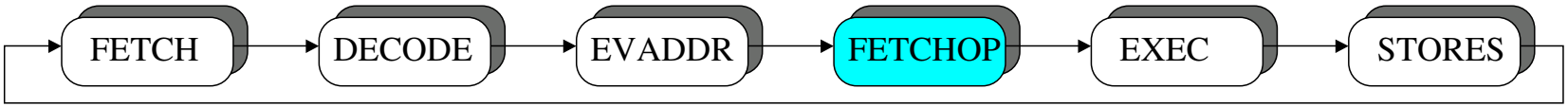
Instruction cycle: EVALUATE ADDRESS



For instructions that require memory access, compute address.

- Different addressing modes are typically supported
  - Indirect mode: Address contained in register.
  - PC-relative mode: Instruction contains offset relative to PC.
  - ...

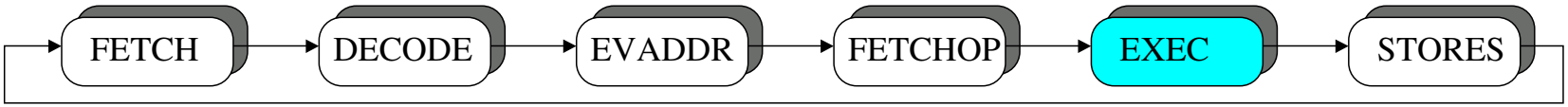
Instruction cycle: FETCH OPERANDS



Obtain source operands for instruction.

- ADD: Always requires two registers as operands. Read registers.
- LDR (read from memory): Load data from memory.

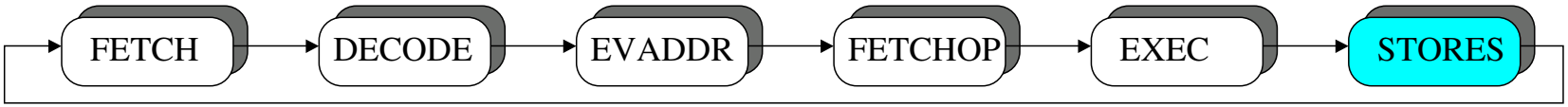
Instruction cycle: EXECUTE



Execute instruction.

- ADD: Use ALU to add source operands.
- Do nothing: e.g. LDR

Instruction cycle: STORE RESULT



Store results.

- ADD: Result of addition stored in register.
- LDR: Data read from memory is stored in register.
- SDR (write to memory): Data is written to memory.