System I

The Processor: Basic Principles

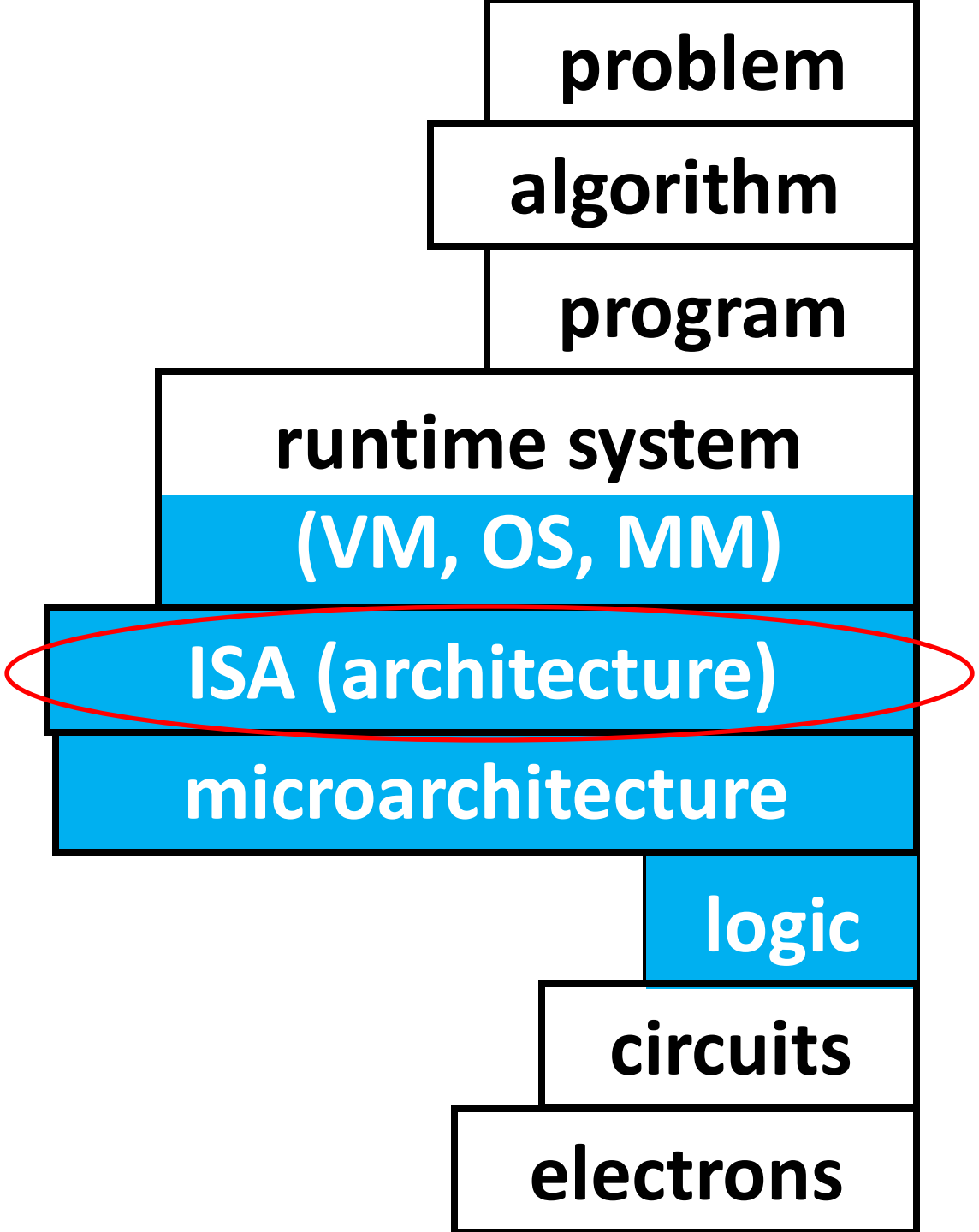
Li Lu

Zhejiang University

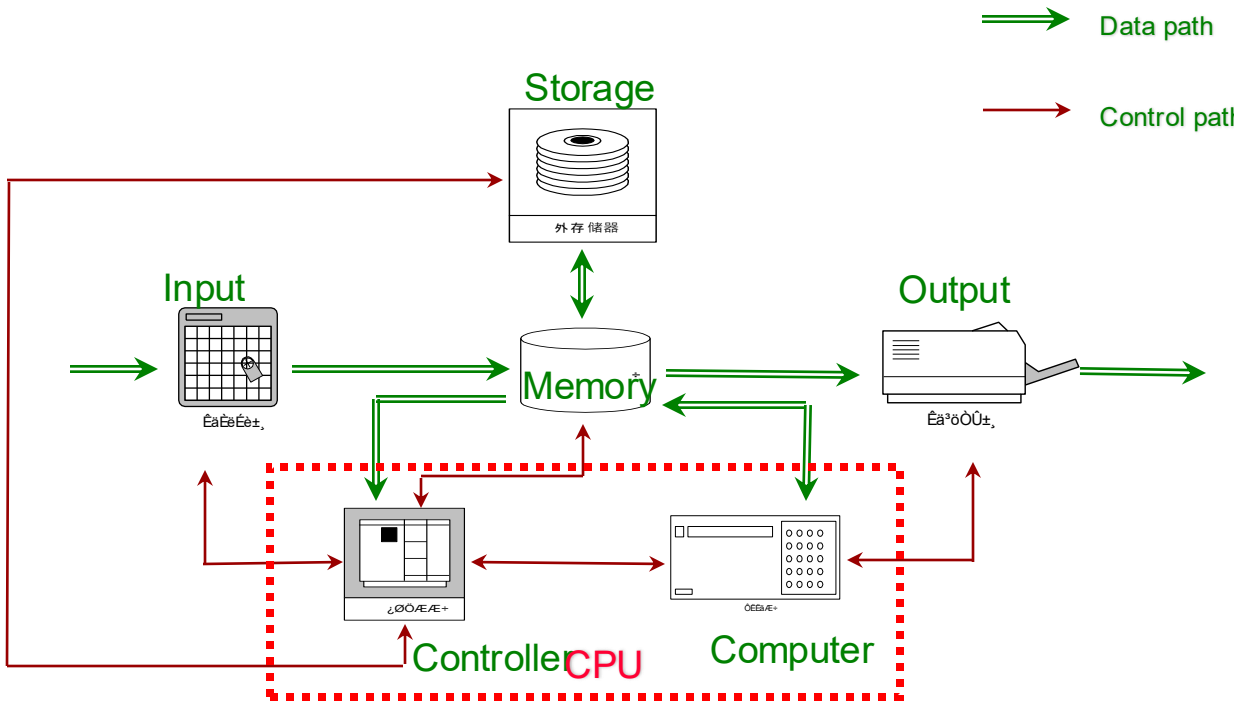
Overview

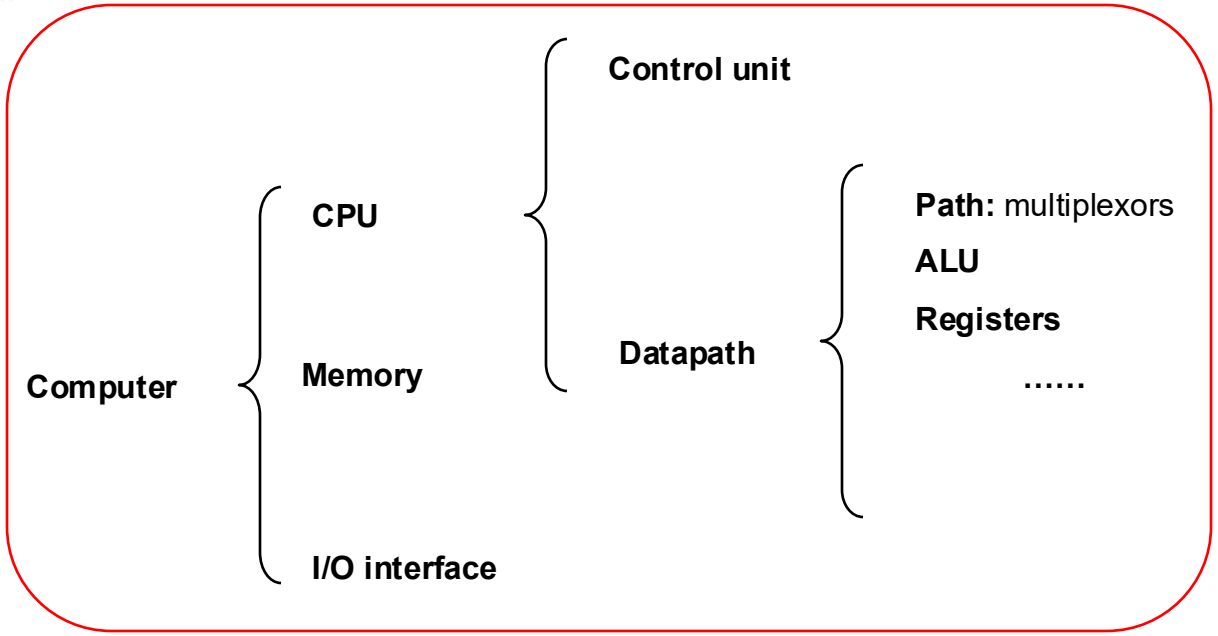
- What is the architecture of computers?
- What is the inside of processor (CPU)?
- How to use CPU to solve problems?

What is the architecture of computers?


problem
algorithm
program
runtime system
(VM, OS, MM)
ISA (architecture)
microarchitecture
logic
circuits
electrons

What is the architecture of computers?





- Von Neumann structure: data and programs are in memory.
- CPU takes instructions and data from memory for operation and puts the results into memory.

Von Neumann Structure

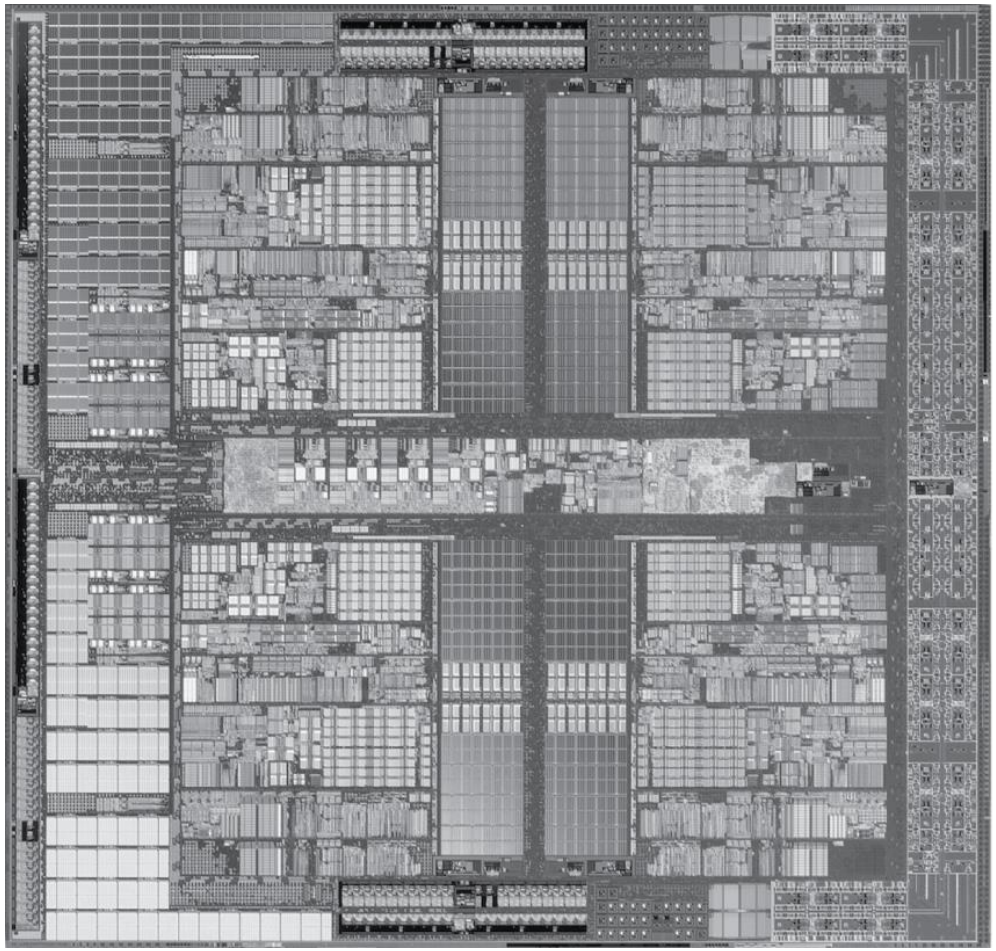
What is the inside of Processor ?

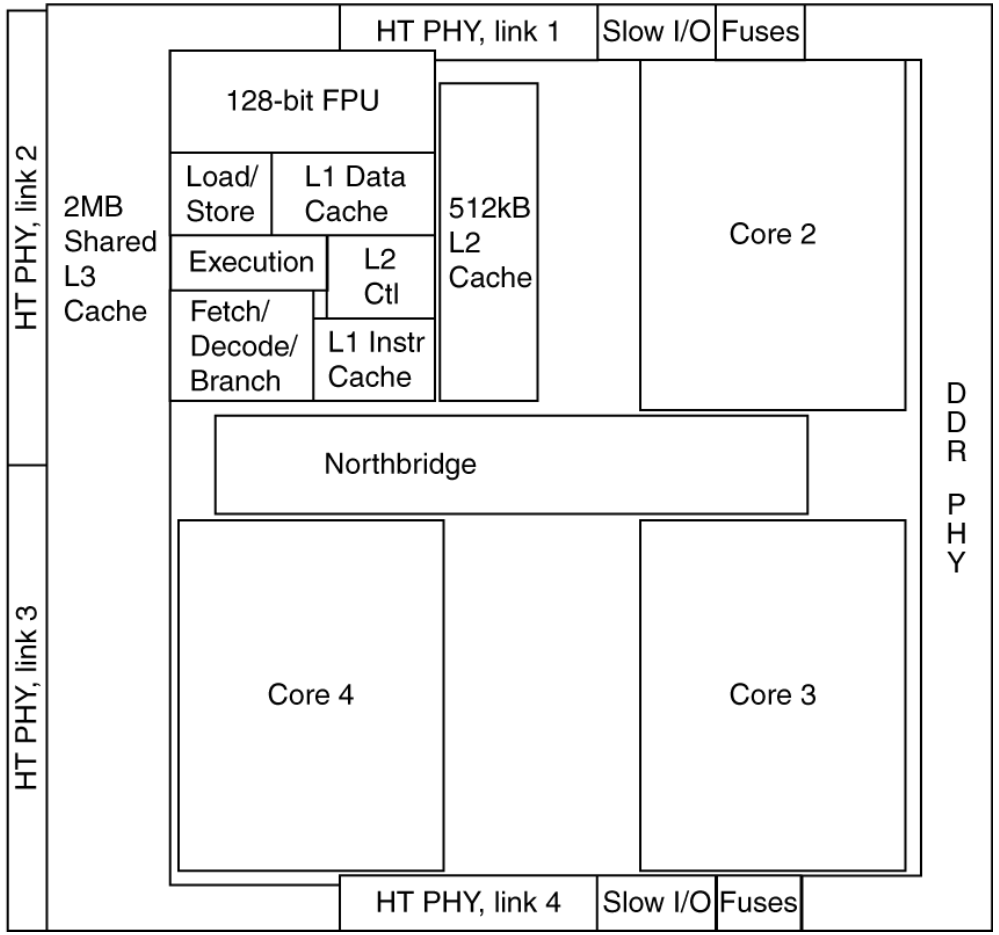
What is the inside of Processor (CPU)?

- Datapath: performs operations on data
- Control: sequences datapath, memory, ...
- Cache memory
  - Small fast SRAM memory for immediate access to data

What is the inside of Processor (CPU)?

- AMD Barcelona: 4 processor cores





What is the inside of Processor (CPU)?

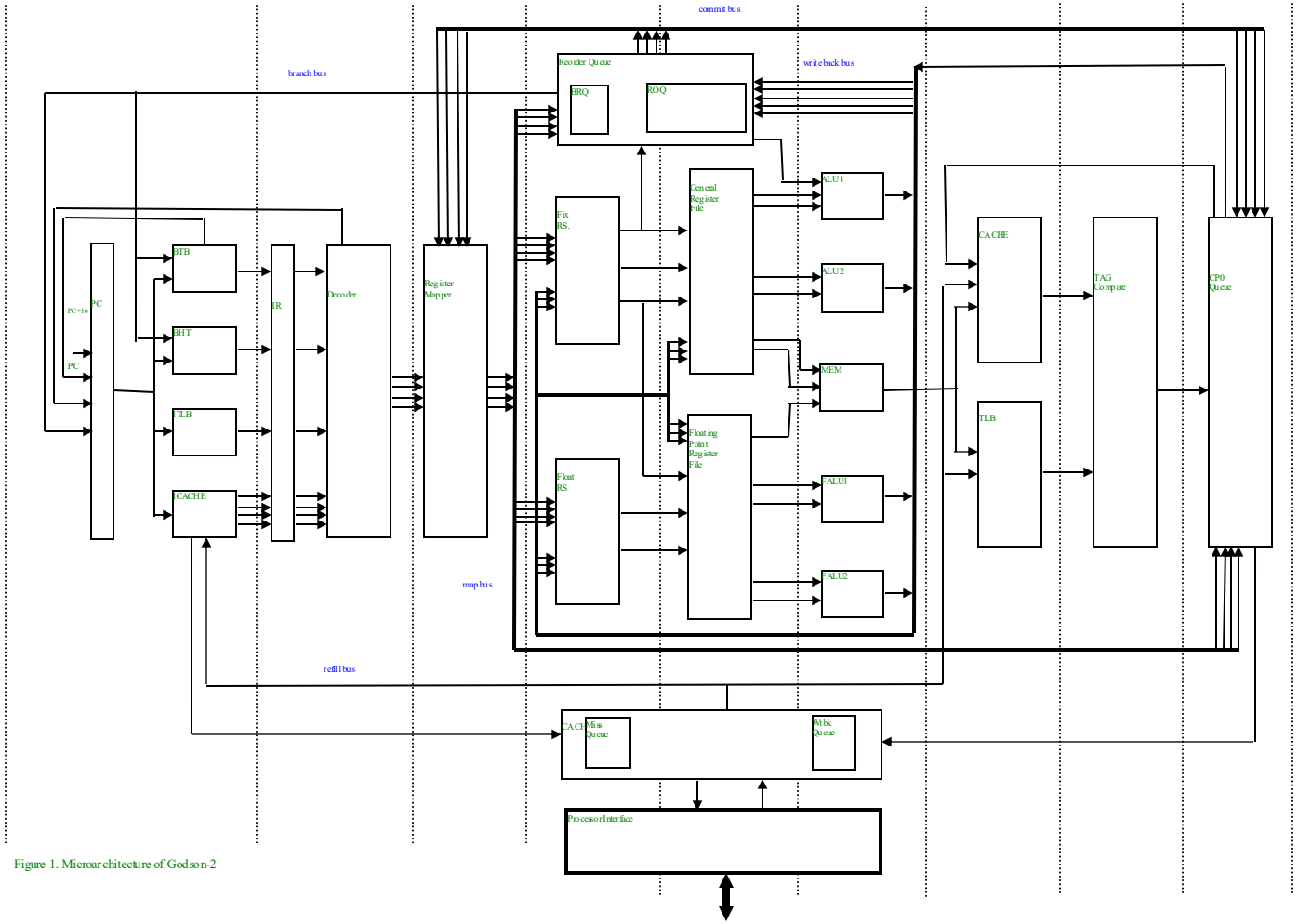
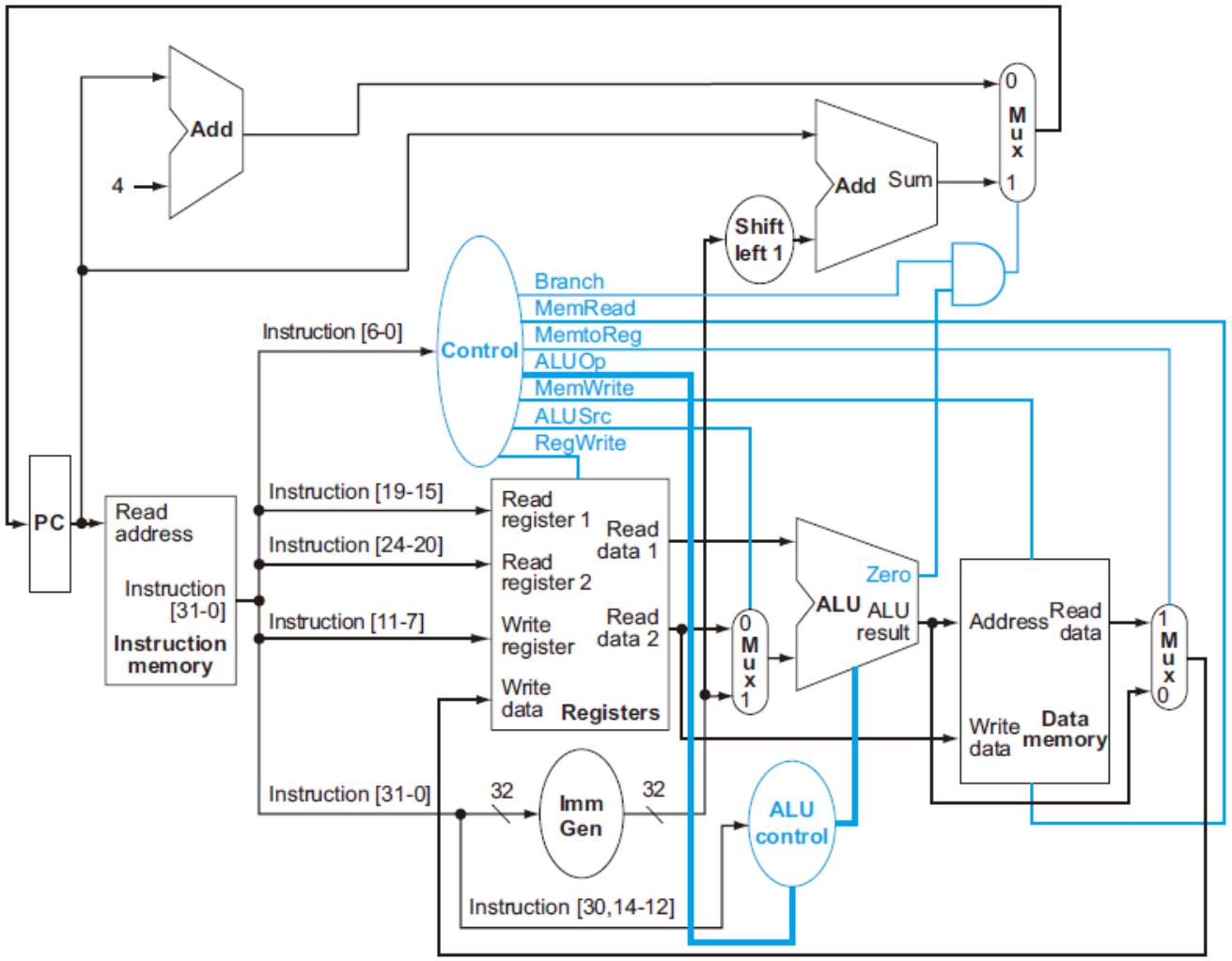


Figure 1. Microarchitecture of Godson-2

A real CPU design

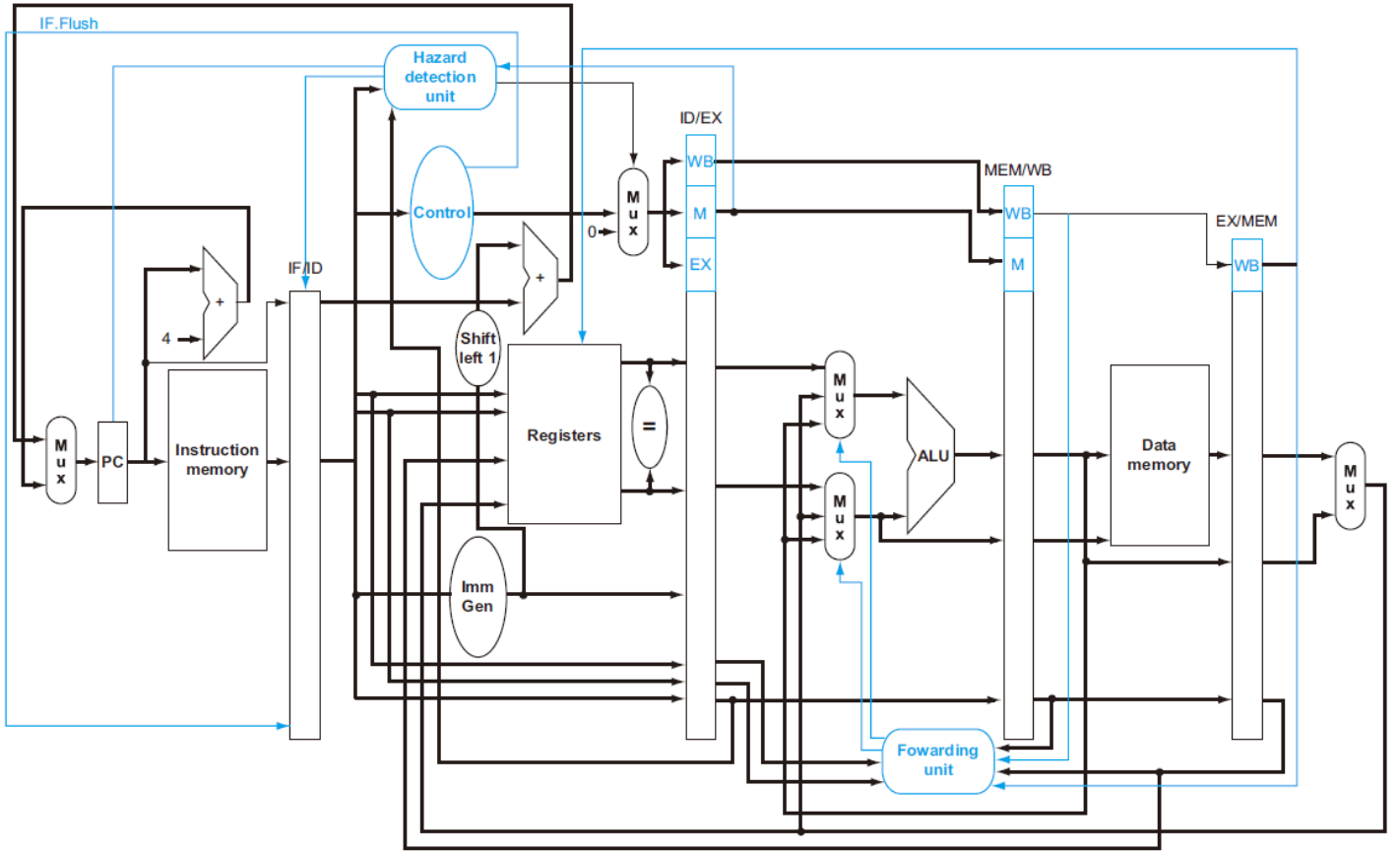
What is the inside of Processor (CPU)?

Datapath with Control



Single cycle
instruction execution

What is the inside of Processor (CPU)?



IF	ID	EX	MEM	WB

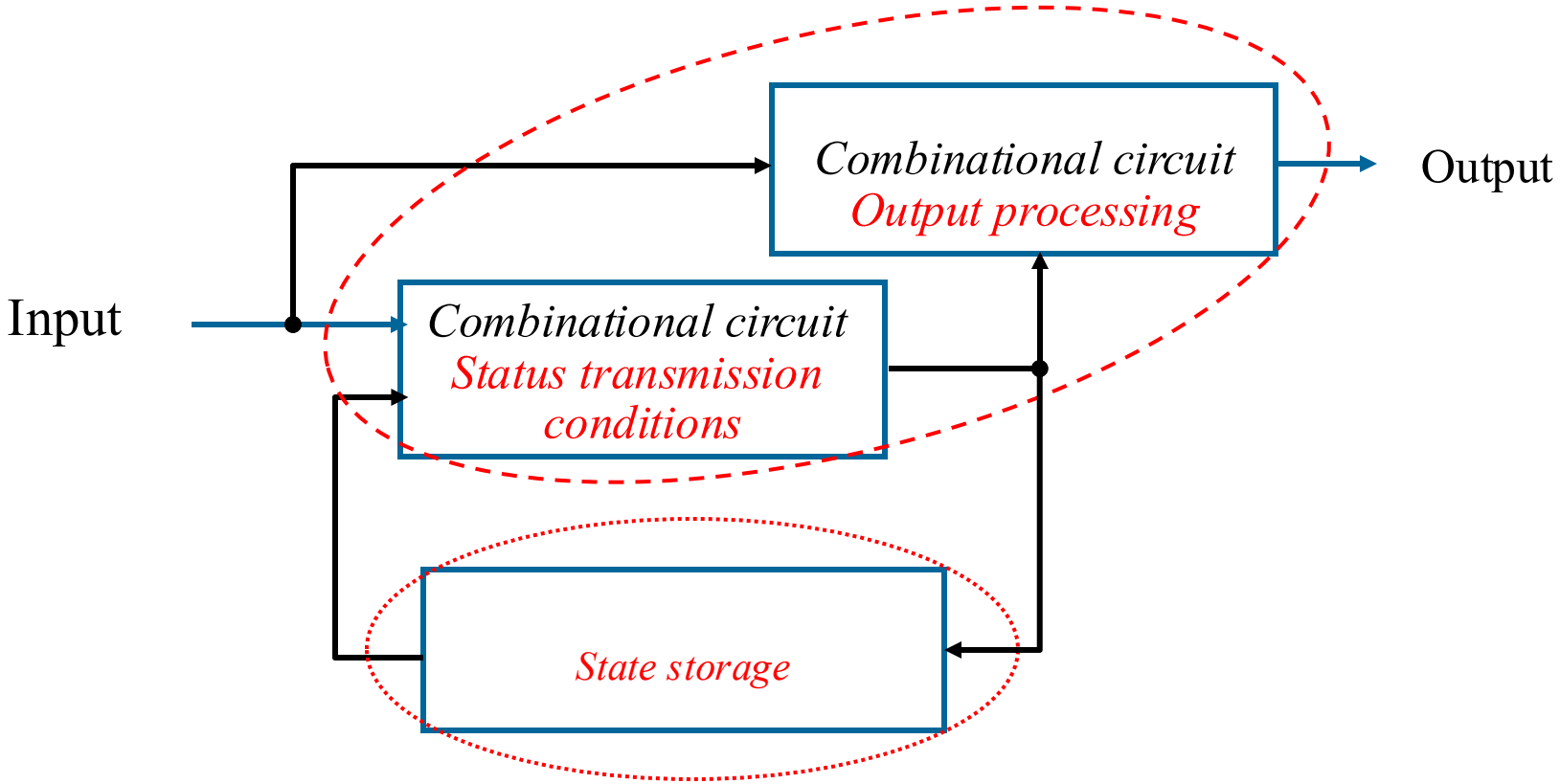
Pipelining

Divide instruction execution into stages

How to use CPU to solve problems?

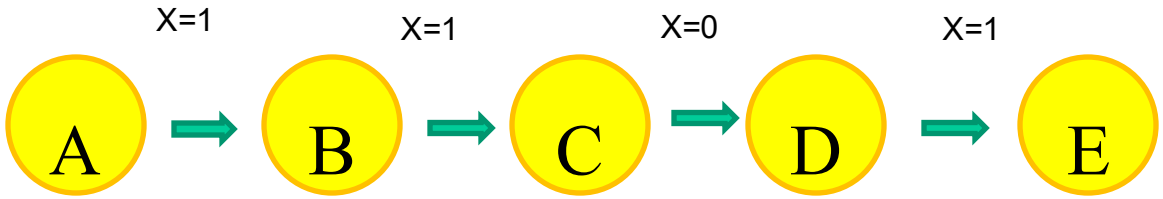
Design methodology 1: Common digital system

◎Finite state machine （FSM）

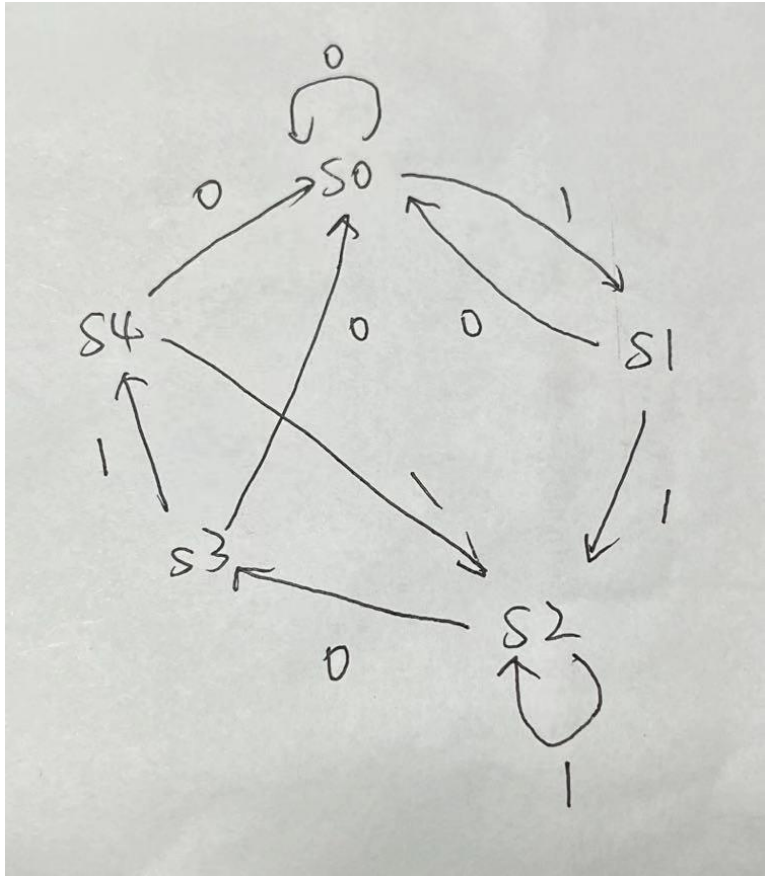


Detection successive occurrence "1101"　number of times

◎How to use FSM ？

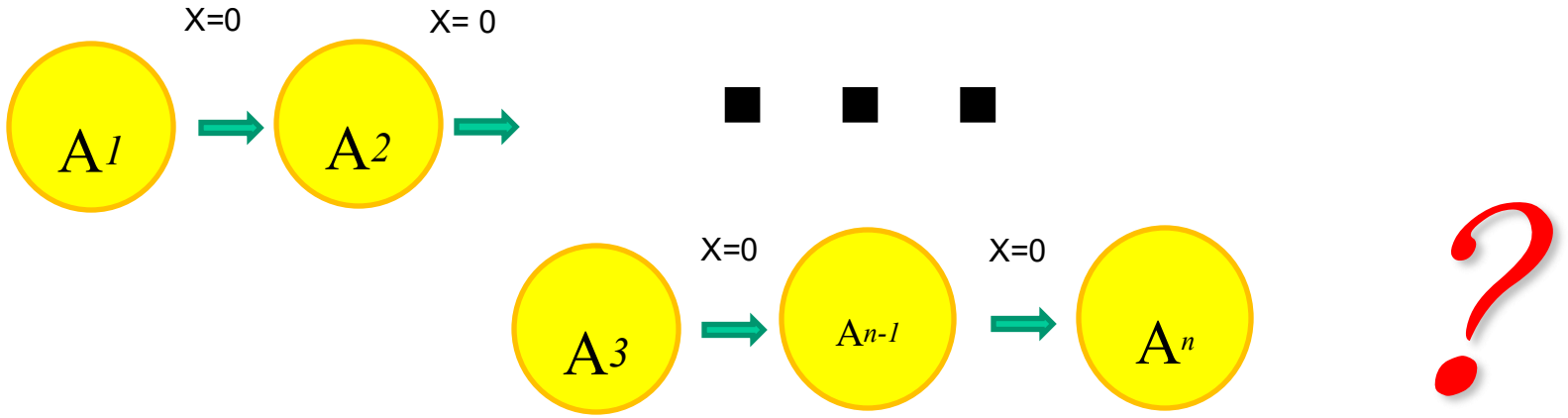


■ ■ ■



Infinite state machine

Detection consecutive "0" number



Impossible to implement

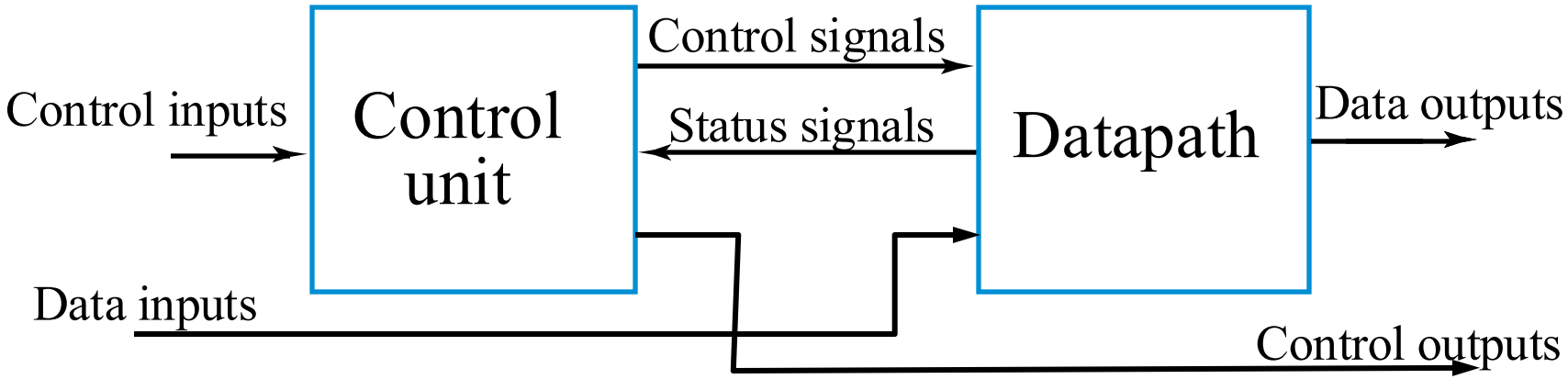
Design methodology 2: Control of Register Transfers

◎Control of Register Transfers

- Register transfers performed on registers control that supervises the sequencing of the register transfers

◎Three essential elements

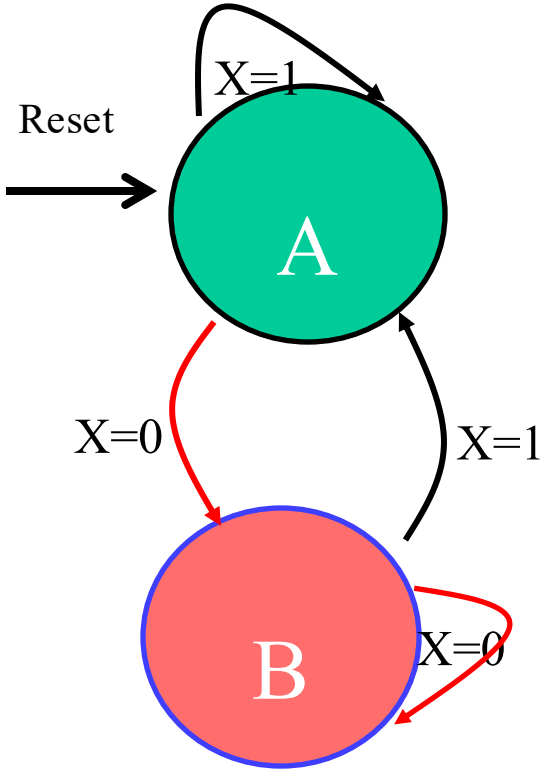
- Set of registers: mostly in Datapath with some in Control Unit
- Basic operation (micromanipulation): Register transfers performed
- Control: that supervises the sequencing of the register transfers

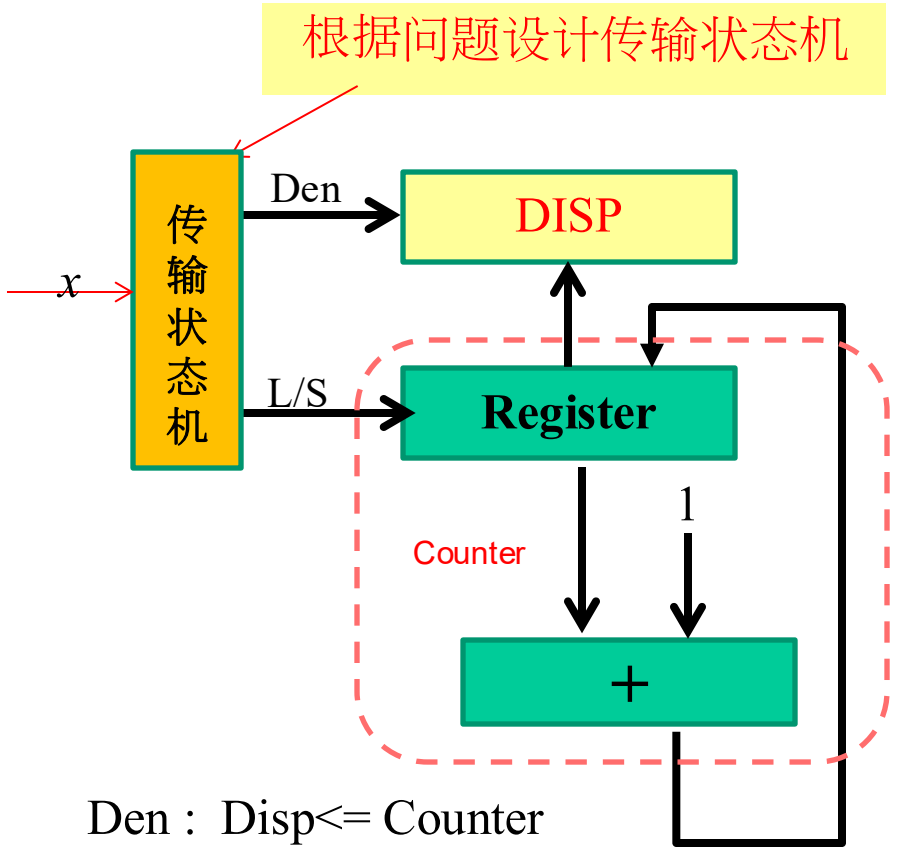


◎Register Transfer Language: RTL

Transmission state machine processing

Detection consecutive "0" number

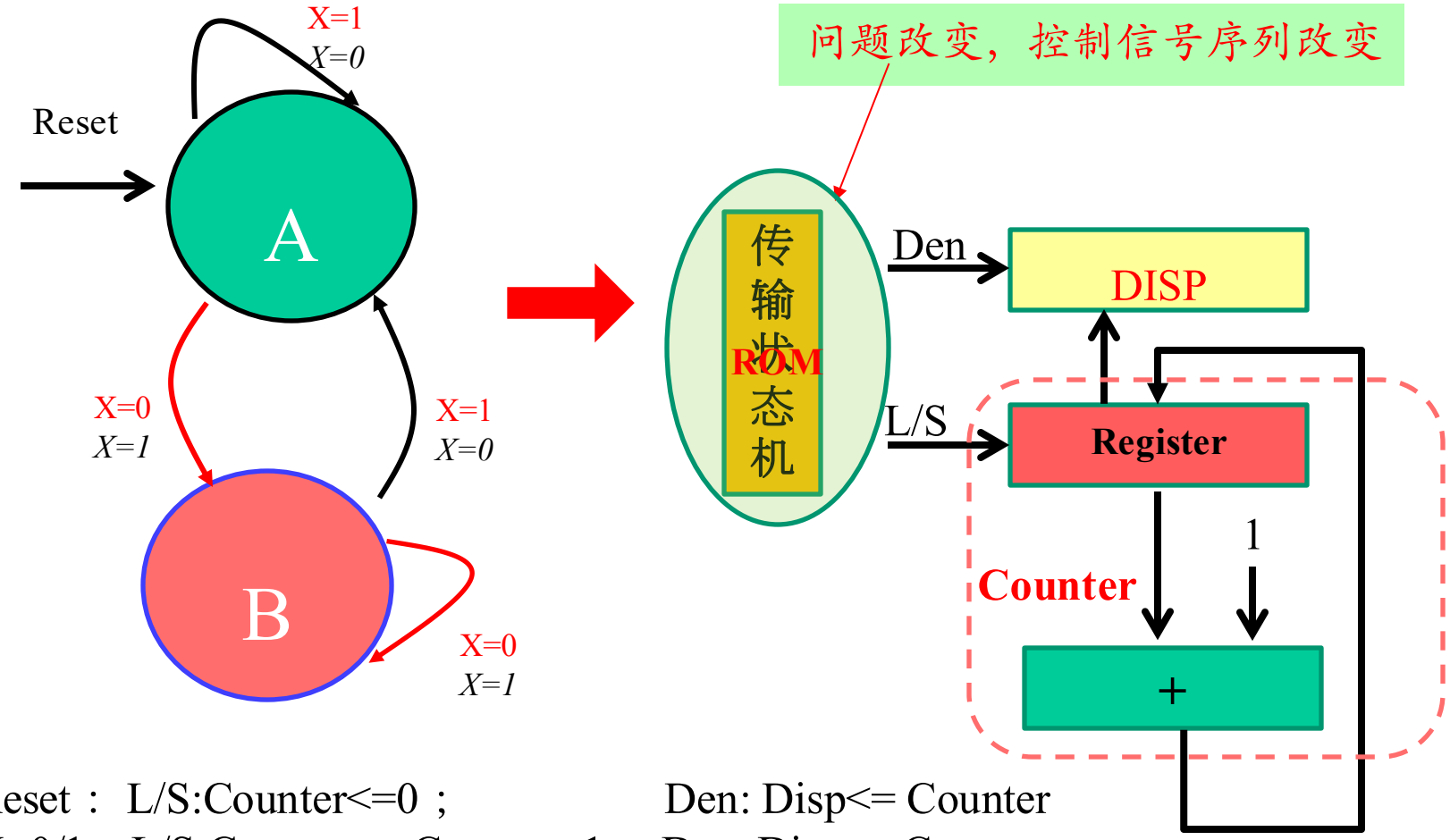




Reset: L/S:Counter<=0 ;
X=0 : L/S:Counter<= Counter+1;
X=1 : L/S:Counter<= Counter ;

Den : Disp<= Counter
Den : Disp<= Counter
Den : Disp<= Disp

Hardware Programmable：Detection consecutive “0/1" number



Reset： L/S:Counter<=0 ;　　Den: Disp<= Counter

X=0/1 ：L/S:Counter<= Counter+1;　　Den: Disp<= Counter

X=1/0 ：L/S:Counter<= Counter ;　　Den: Disp<= Disp

Specific and general system

◎Non-programmable System: Specific System

- The control unit does not deal with fetching and executing instructions
- But contains all of the information for sequencing register transfers based on inputs and on status bits from the datapath

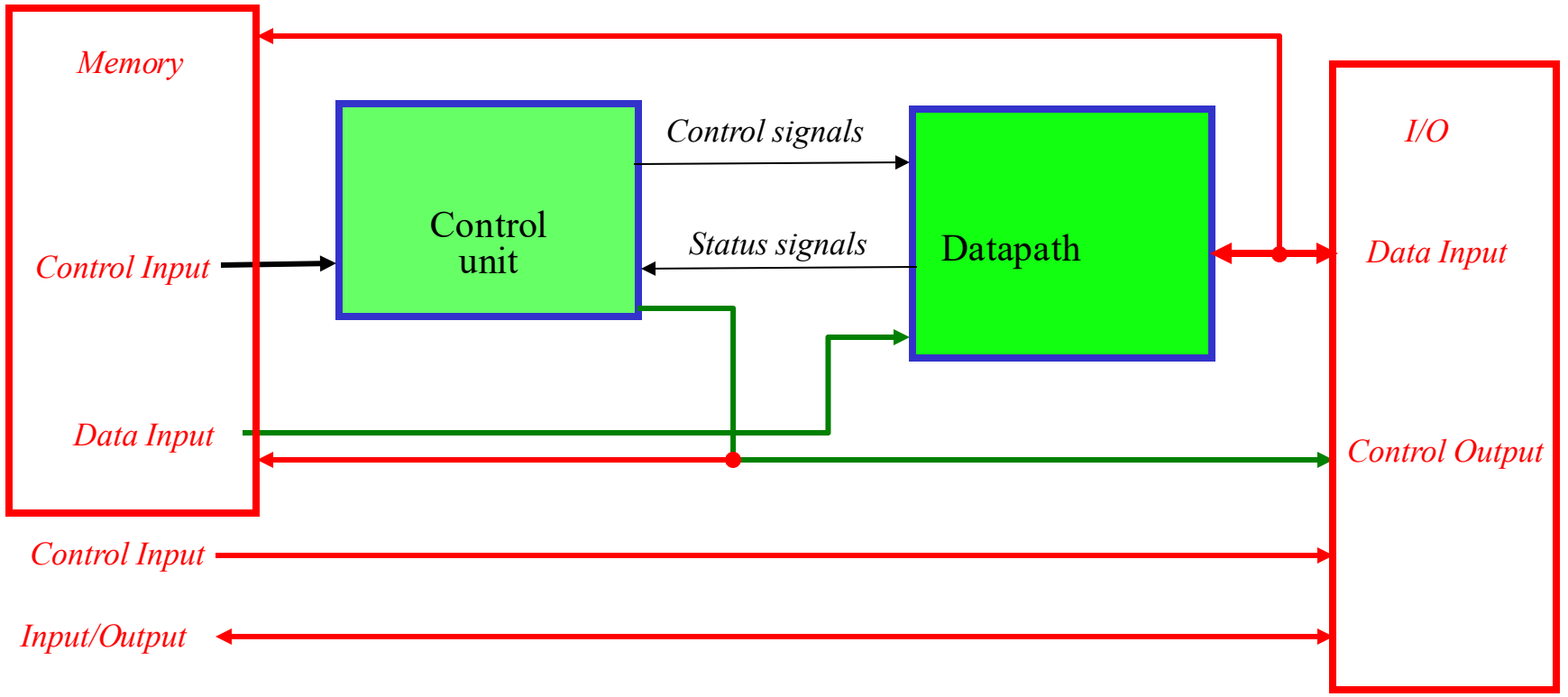
◎Programmable System : General system

- A portion of the input consists of a sequence of *instructions* called a *program*,
- Typically stored in a memory and addressed by a *program counter*.
- The Control Unit is responsible for fetching and executing these instructions.

Design methodology 3: Control of Register Transfers

◎General system

- Program state machine （PSM）



Is there a better way?

--Universal Turing machine thinking

Turing Machine

◎Basic idea

- Use machine to simulate mathematical computation process of humans by paper and pen
- Two simple operations
  - Write or delete a symbol from the paper
  - Move the attention from one position to another of the paper
- The next operation depends on
  - (a) the symbol in one position of the paper that the person pays attention to
  - (b) the thinking state of the person currently
- Turing constructs a conceptual machine

Components of Turing Machine

1. An infinitely long paper tape TAPE

   The paper is divided into several consecutive blanks, each of which contains a symbol from a finite set (the alphabet). The alphabet includes a special symbol □ representing the empty. The blanks in the tape is numbered as 0, 1, 2, …, from the left to right. The right end of the tap can be extended infinitely

2. A read/write head HEAD

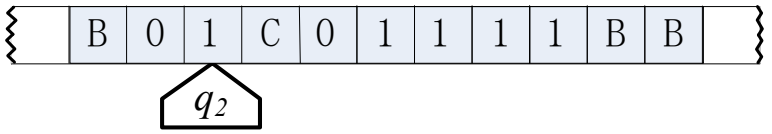
   The head can move left or right on the tape, which can read the symbol of the current blank, and is able to modify the symbol of current blank

3. A set of control rules TABLE

   It can determine the next operation of the head based on current state of the machine and the symbol under the head currently, and modify the value of state register, leading the machine to a new state

4. A state register

   Used to save the current state of Turing Machine



Turing Model：Marvin • Minsky (1967)

Instruction structure of Turing Machine

◎Typical seven-tuples set (has many variants)

- **M={Q, Σ, Γ, δ, q0, qaccept, qreject}**, where Q, Σ, Γ are all finite sets

1. Q is the state set;
2. Σ is the input alphabet, which does not include the special empty "□";
3. Γ is the alphabet, where b∈Γ is the empty, and Σ∈Γ;
4. δ is the transfer function: $Q \times \Sigma \rightarrow Q \times \Gamma\{L,R\}$, where L and R represent the read/write head moves left and right, respectively;
5. q0∈Q is the beginning state;
6. qaccept is the accept state;
7. qreject is the reject state, and qreject≠qaccept.

Program of Turing Machine

◎State Transfer Function

- Program

  δ={ qi Sj Sk R(L or N) ql }

  qi is the current state of the machine;

  Sj is the symbol read by the machine;

  Sk is the symbol waiting to be written to replace Sj;

  R, L, N represent moving right one step, moving left one step, or remain static;

  ql is the next state of the machine

- Example: δ={q0 0 0 R q0; q0 1 0 R q0}
  - Move right and delete all blanks that passed by the head in the tape
  - Never stop and move backward

Turing Machine Program δ: Consecutive 1 Detection

◎7 inner states except for the beginning state q0

q0 B B R q1	//起始状态,读写头指向左边缘,次态为q1。
q1 1 C L q2	//状态1,检测到“1”,标记C,开始计数。
q1 0 C L q7	//状态1,检测到“0”,标记C,开始清零。
q1 B B N q1(STOP)	//状态1,检测到右边缘停机,左边是检测结果。
q2 0 1 L q4	//状态2,计数。从右（低位）向左（高位）计数。
q2 1 0 L q2	//状态2,计数。从右（低位）向左（高位）计数。
q2 B 1 L q4	//状态2,到左边缘计数结束，次态是q4。
q4 1 1 L q4	//状态4,读写头移到左边缘。
q4 0 0 L q4	//状态4,读写头移到左边缘。
q4 B B R q3	//读写头到左边缘,开始计数值右移1位。
q3 1 0 R q5	//计数值将右移1位，左边填“0”。
q3 0 0 R q6	//计数值将右移1位，左边填“0”。
q3 C 0 R q1	//右移结束，清计数标志。
q5 1 1 R q5	//状态5，右移“1”。
q5 0 1 R q6	//状态5，右移“1”。
q5 C 1 R q1	//右移“1”并结束右移，清计数标志。
q6 1 0 R q5	//状态6，右移“0”。
q6 0 0 R q6	//状态6，右移“0”。
q6 C 0 R q1	//右移“0”并结束右移，清计数标志。
q7 1 0 L q7	//状态7，清零。
q7 0 0 L q7	//状态7，清零。
q7 B B R q3	//检测到左边缘，清零结束。转到状态3，计数值右移一位。

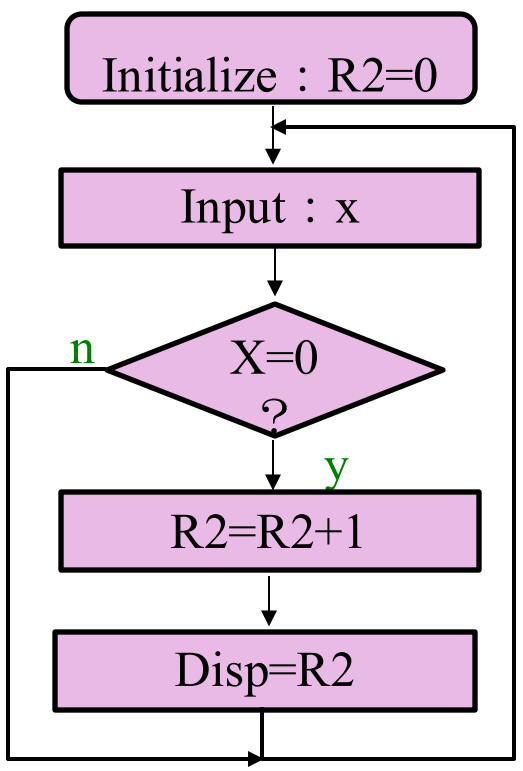
Universal Turing Machine Implementation

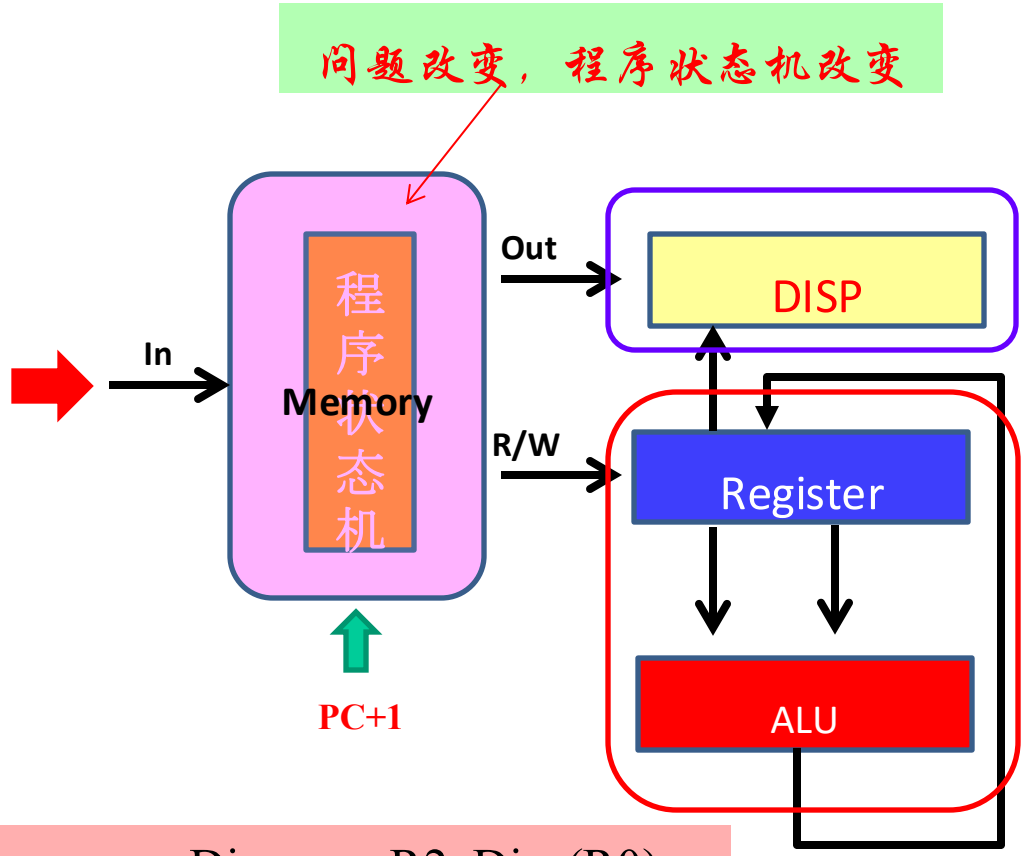
- Ideal Turing Machine cannot be implemented
  - Cannot find an infinite long tape
    - Not universal computing system neither
    - Only fixed and mechanical computing system
- Universal (General) Turing Machine
  - Long enough tape with head and tail connected, to replace the infinite long paper tape
  - Any Turing machine instruction code
    - Write on the beginning part of the tape
  - Operate on the rest part of the input
  - Still low efficiency
- Two problems
  - Carrier
  - Efficiency

Practical Turing Machine — A Real Computer

- A large enough storage to replace the tape
  - Magnetic storage for replacement
  - Semiconductor storage for replacement
- Storing the inner state
  - Inner state does not store immediate, downgrading the efficiency
  - Register to store the immediate (i.e., subset of tape)
  - Can significantly simplify computational structure and improve computing efficiency
- Enhance the processing capability of HEAD--CPU
  - Require a stronger HEAD
  - Can be implement by Control of Register Transfer Technique: CPU
  - Satisfactory Universal Turing Machine

Programable Register Transfer Technology to Implement the Universal Turing Machine






```
Reset:    add R2, R0,R0 ;          Disp : sw R2, Disp(R0);
Input:    lw R3, InPort(X);
X=0 :     addi R2, R0,1 ;          Disp : sw R2, Disp(R0);
X=1 :     nop ;
```

Conversion of thinking: from digital systems turn to computer systems

- Basic idea of implementing Turing machine by CPU
  - General digital system
    - Storage control sequence: solving the infinite tape
    - Triple let Instruction: Basic operation of HEAD
    - Control flow instructions: More control rules TABLE
    - Register: More state storage and immediate
  - Computation Thinking: Program solving
    - Algorithm design: Computing thinking
    - Program design: Software thinking
    - Program rules: Instruction→Rule and processing→Control of instructions
      - Define instruction format: Design DataPath and Controller
      - Define data structure: Allocate storage space (Memory、I/O)
  - Software-Hardware Cooperation Processing Thinking

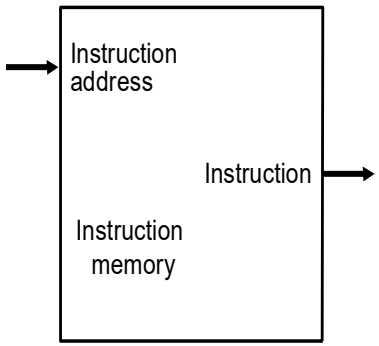
A good implementation of Turing Machine

Turn to use CPU for digital information processing

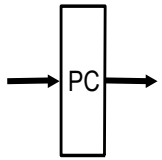
Implementation of CPU

- CPU
  - General digital system
  - A Turing Machine
    - Implemented by Register Transmission Control Technology
      - Datapath
        - The component of processor that performs arithmetic operations
      - Control
        - The component of processor that commands the datapath, memory, and I/O device according to the instructions of the program

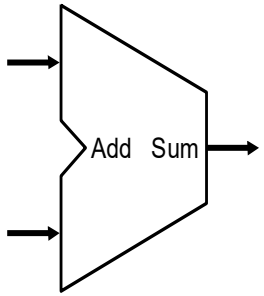
Components in CPU



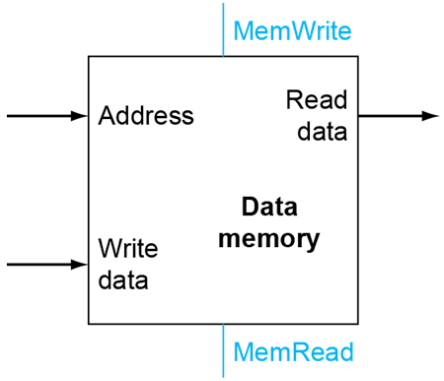
a. Instruction memory



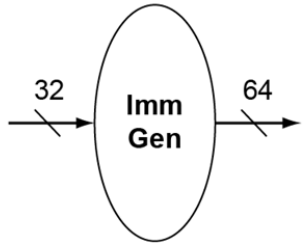
b. Program counter



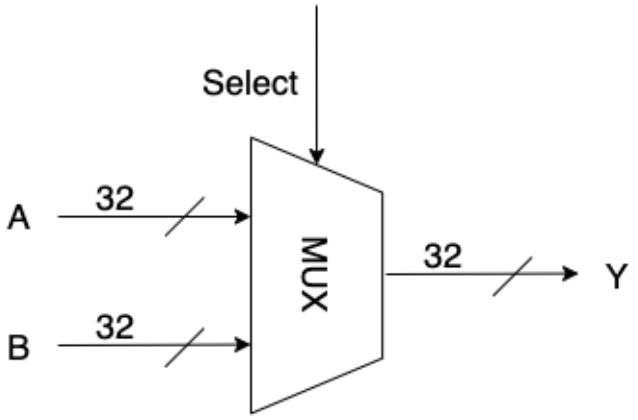
c. Adder



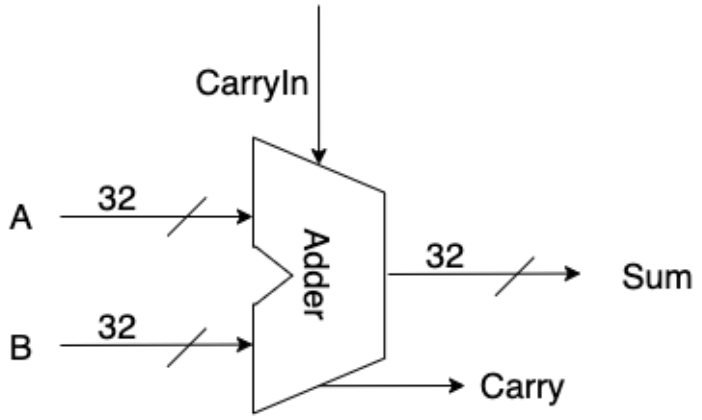
a. Data memory unit



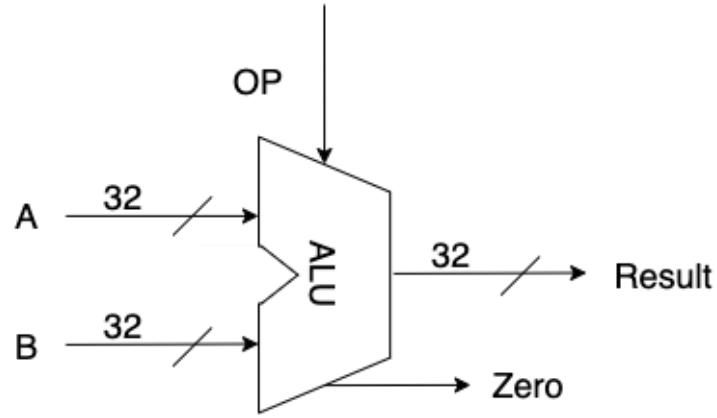
b. Immediate generation unit



Multiplexer



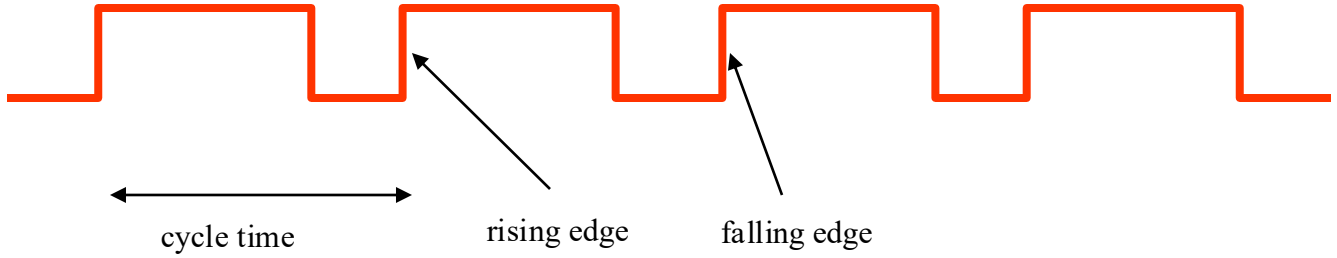
Adder



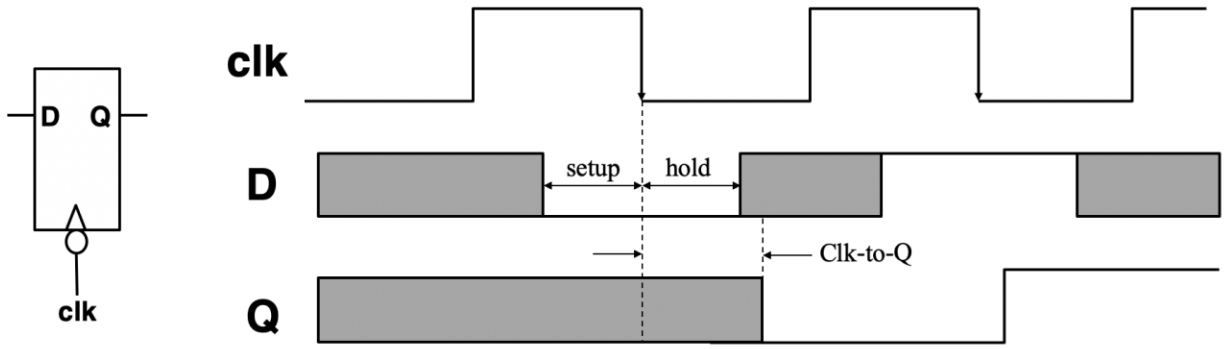
ALU

Logics in CPU

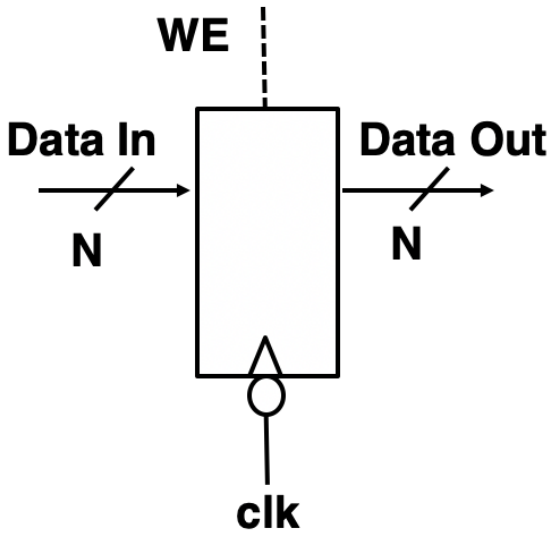
- Unclocked vs. Clocked
- Clocks used in synchronous logic
  - when should an element that contains state be updated?



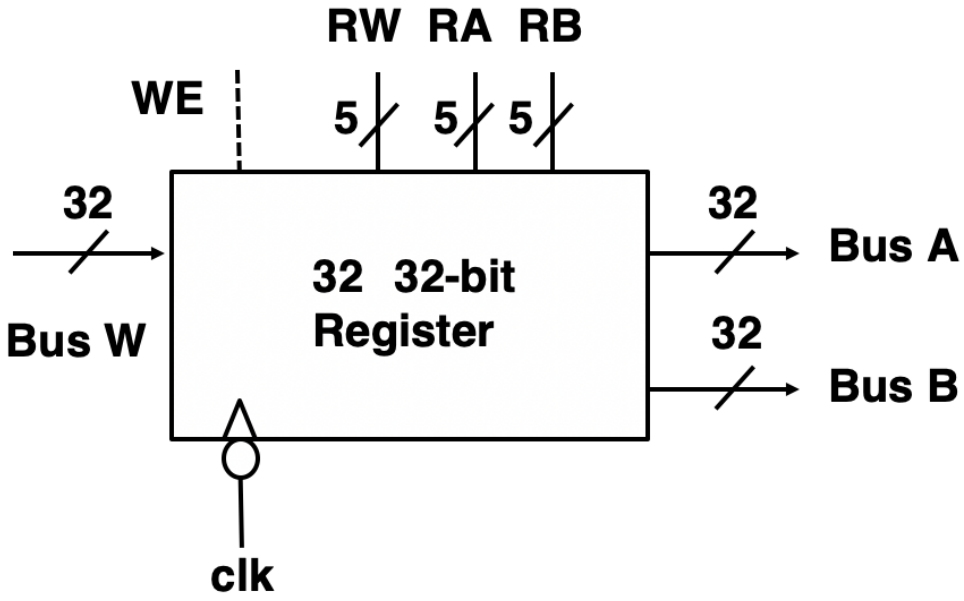
State Elements



State Storage in CPU

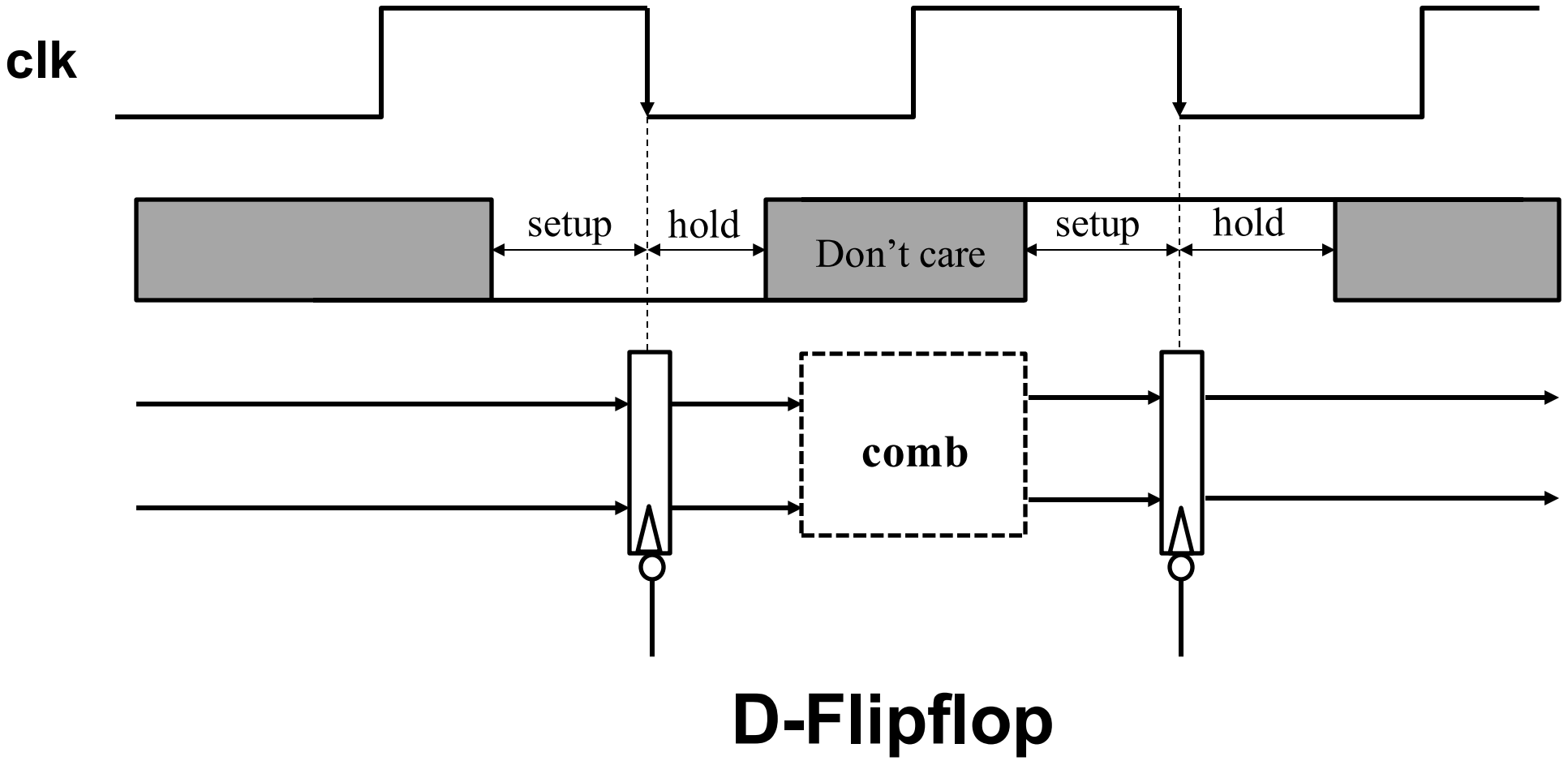


Temporary register



General registers

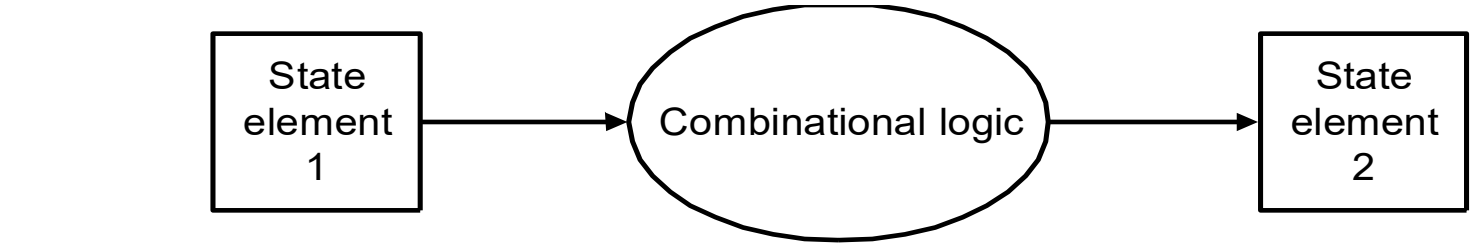
Sequential Logic in CPU



D-Flipflop

Sequential Logic in CPU

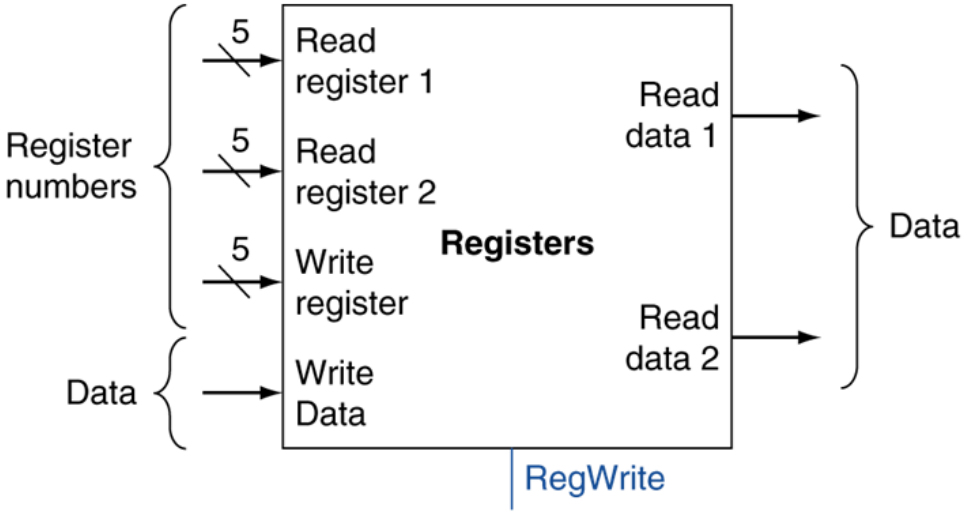
- An edge triggered methodology
- Typical execution:
  - read contents of some state elements
  - send values through some combinational logic
  - write results to one or more state elements

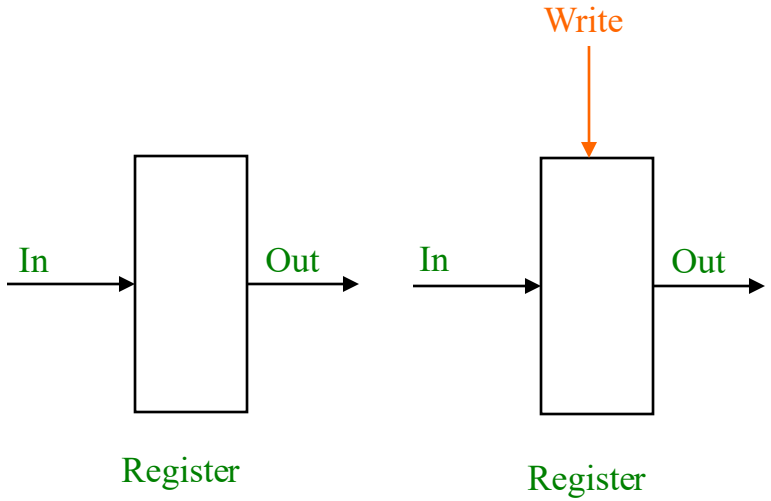


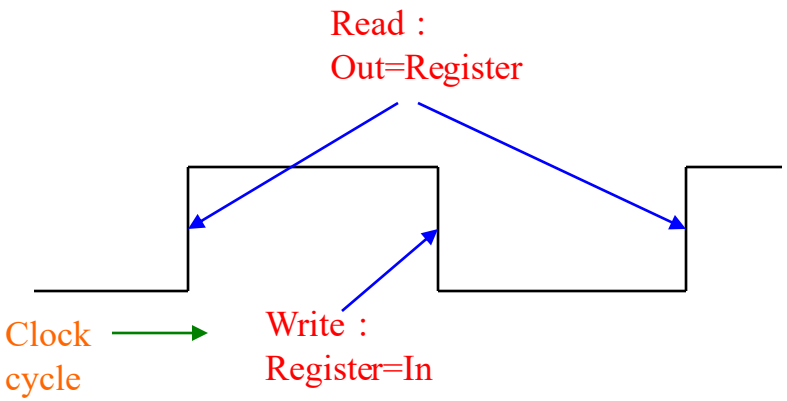
Our Implementation

Sequential Logic in CPU

- Register
  - State element
  - Can be controlled by Write signal







REGISTER

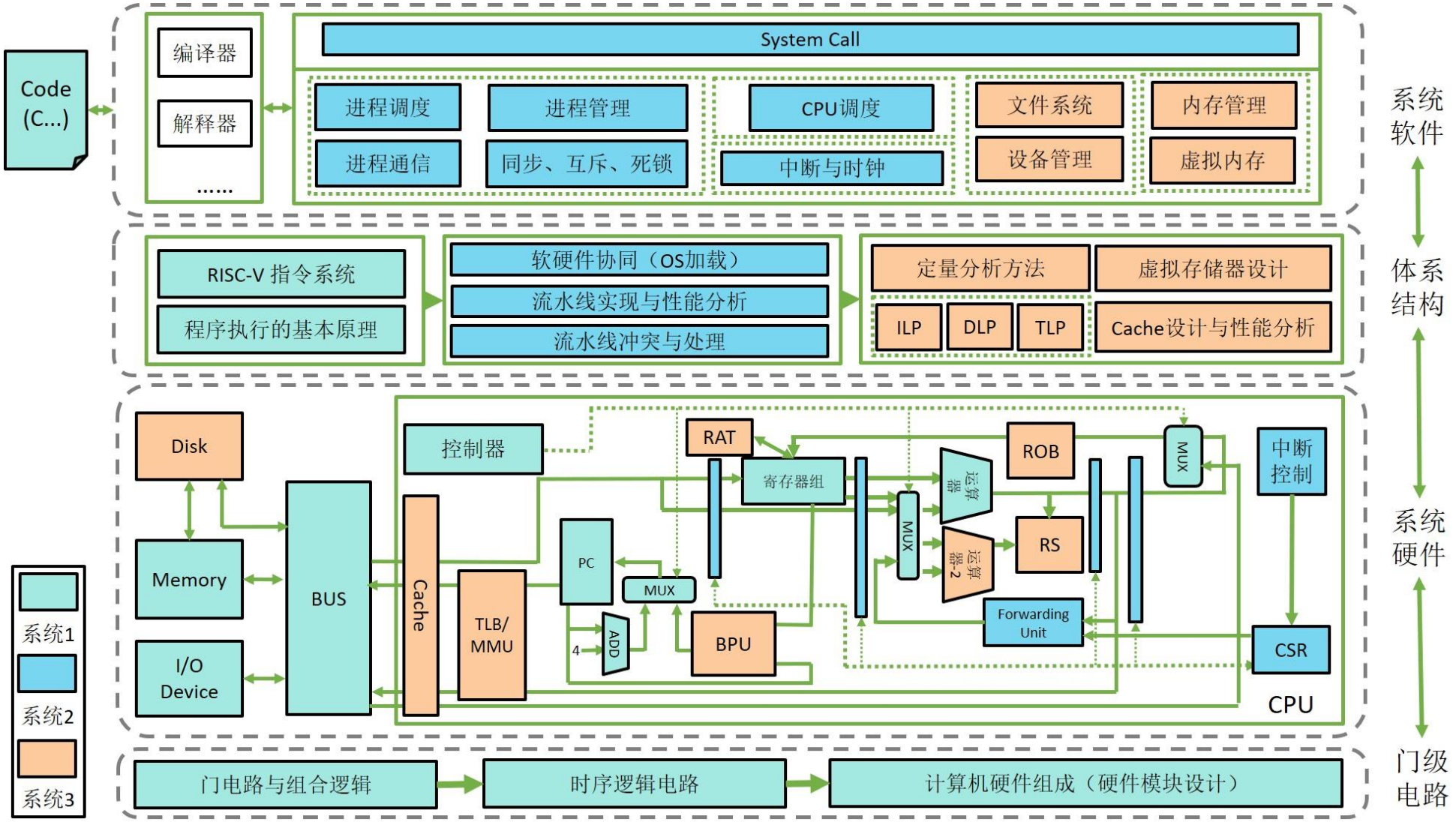
CPU we will be doing

- We'll look at an implementation of RISC-V
- Simplified to contain only:
  - memory-reference instructions: **lw, sw**
  - arithmetic-logical instructions: **add, sub, and, or, slt**
  - control flow instructions: **beq, jal**
- An overview of the implementation
  - For every instruction, the first two step are identical
    1. Fetch the instruction from the memory
    2. Decode and read the registers
  - Next steps depend on the instruction class
    - **Memory-reference** **Arithmetic-logical** **branches**

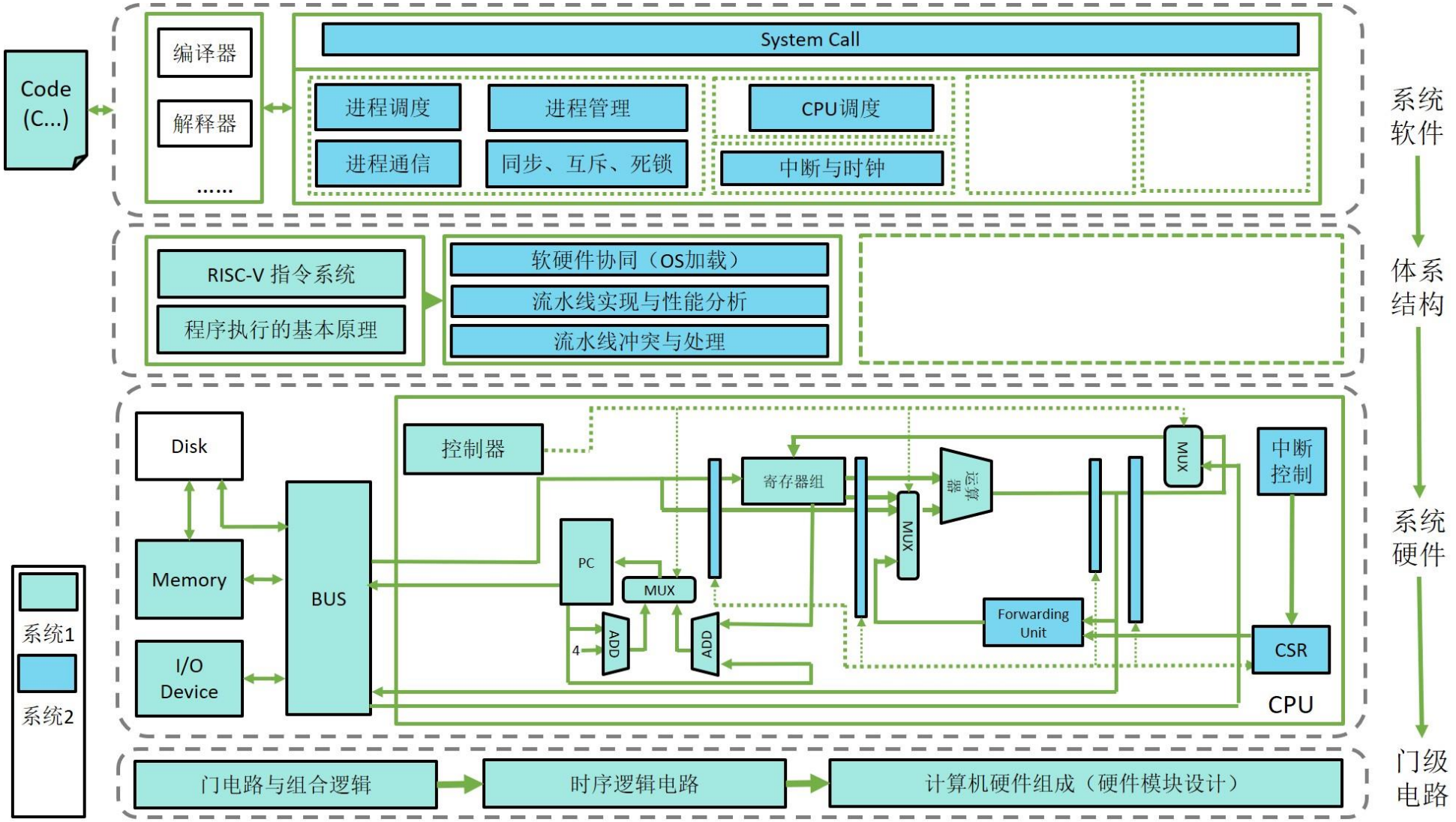
What are the steps?

How many functions

CPU we will be doing



CPU we will be doing



CPU we will be doing