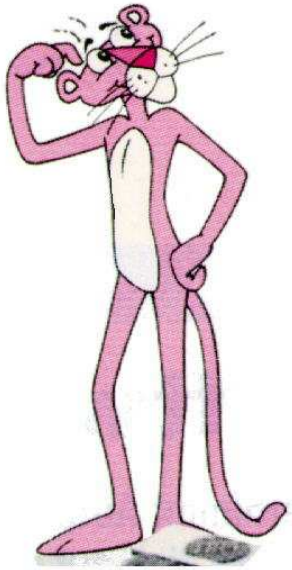
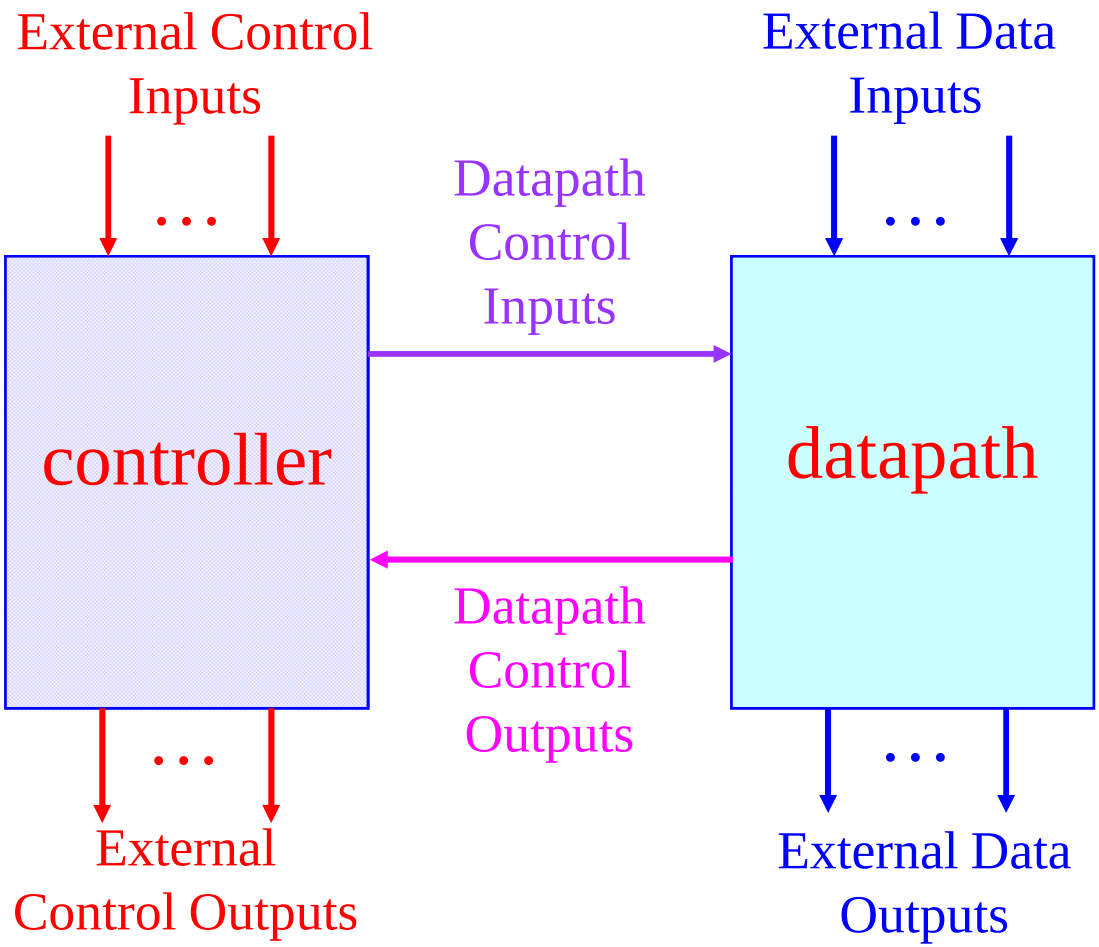
Repeat

Design of Control Units

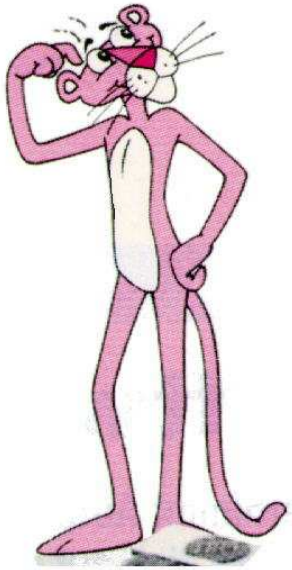
1. Single-Purpose Processor Basic Model

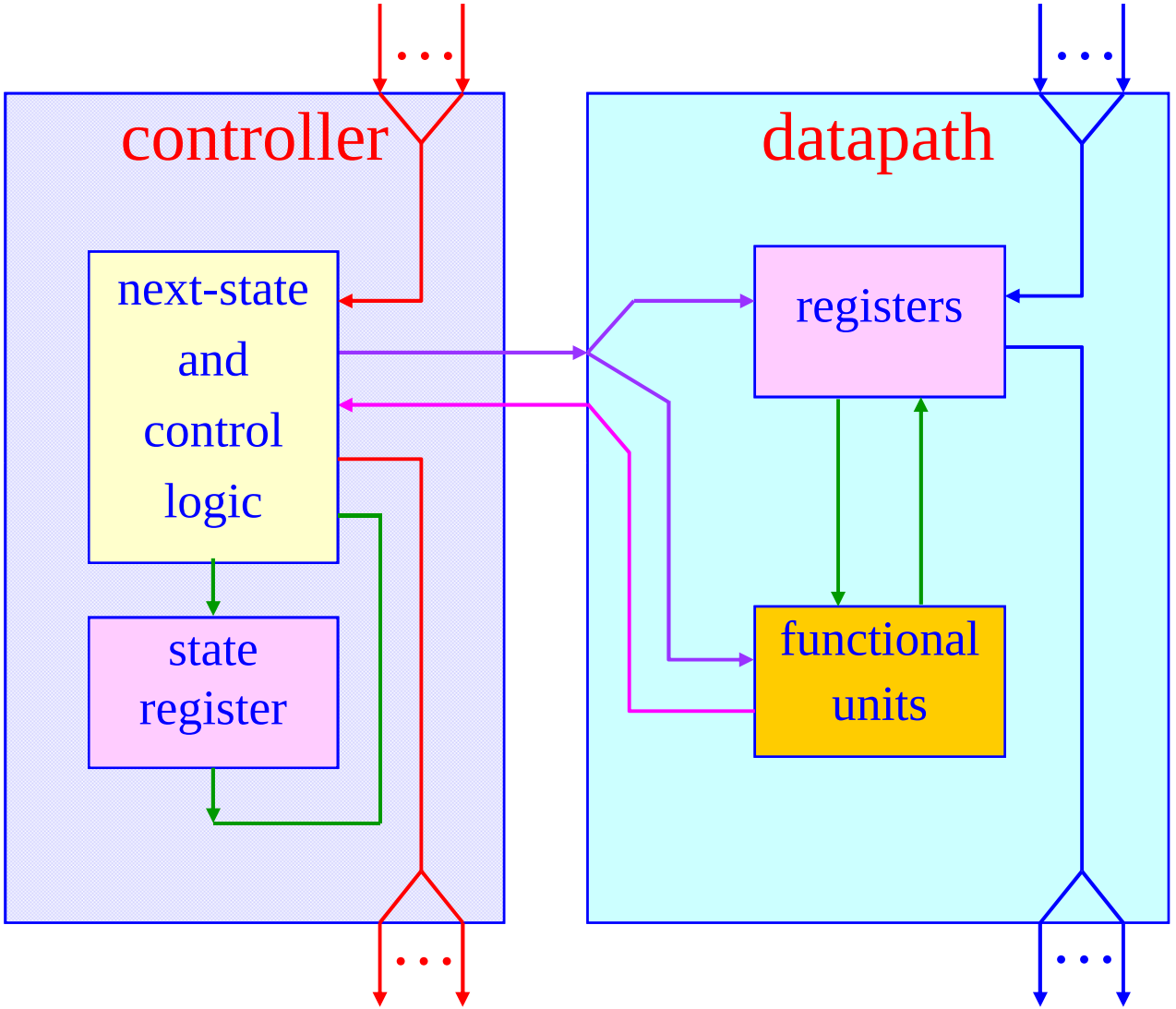




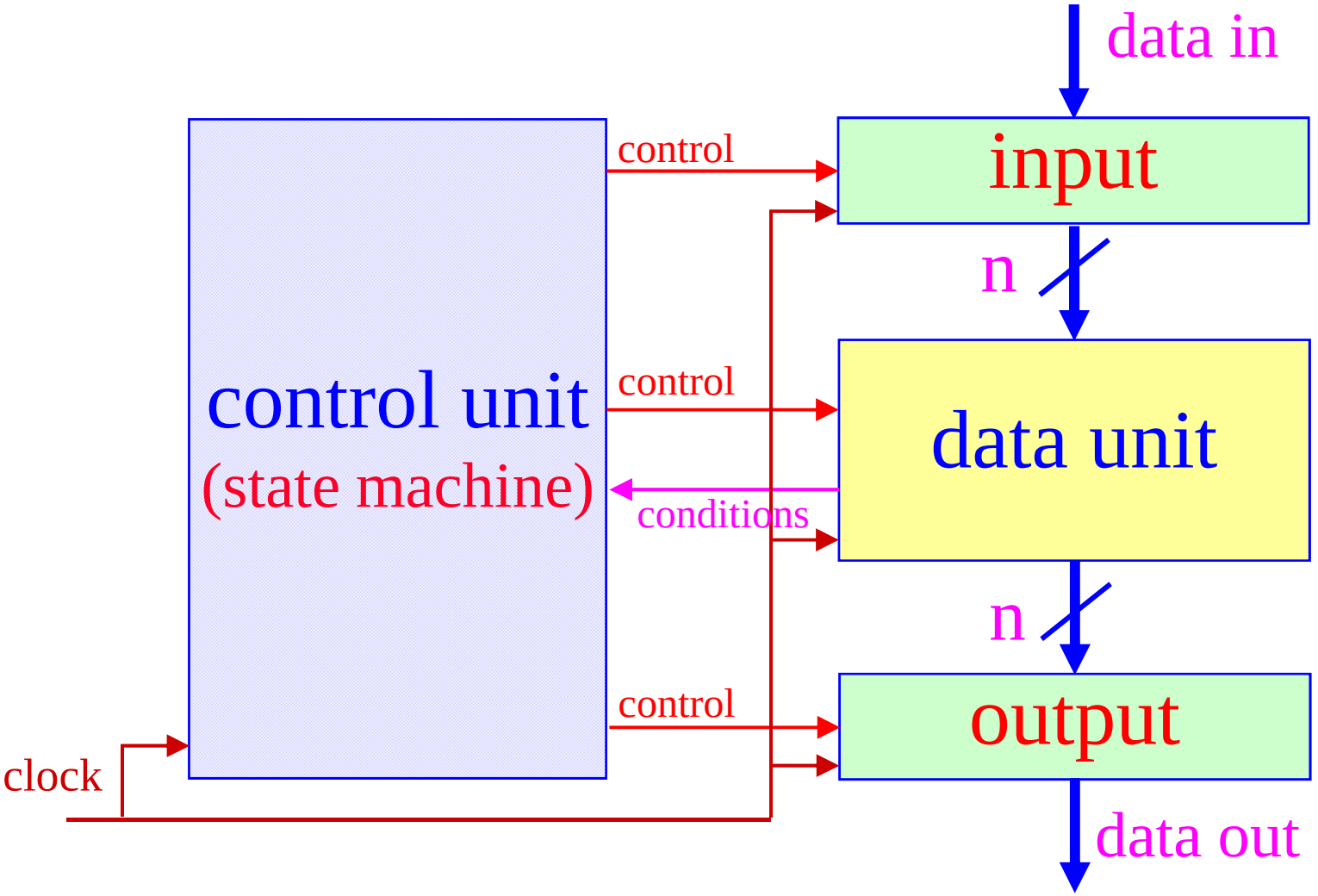
Controller and Datapath

1. View inside: controller and datapath

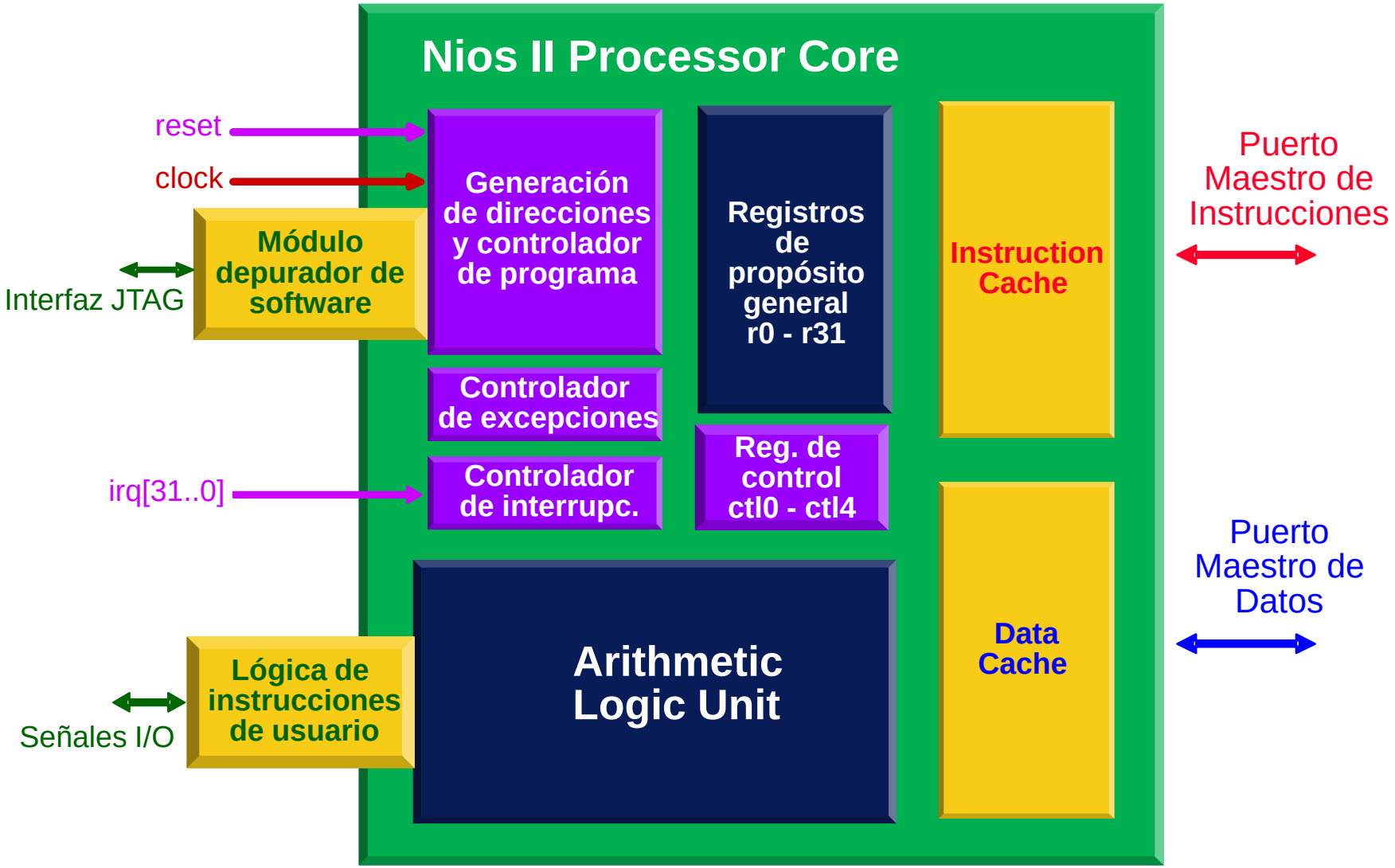


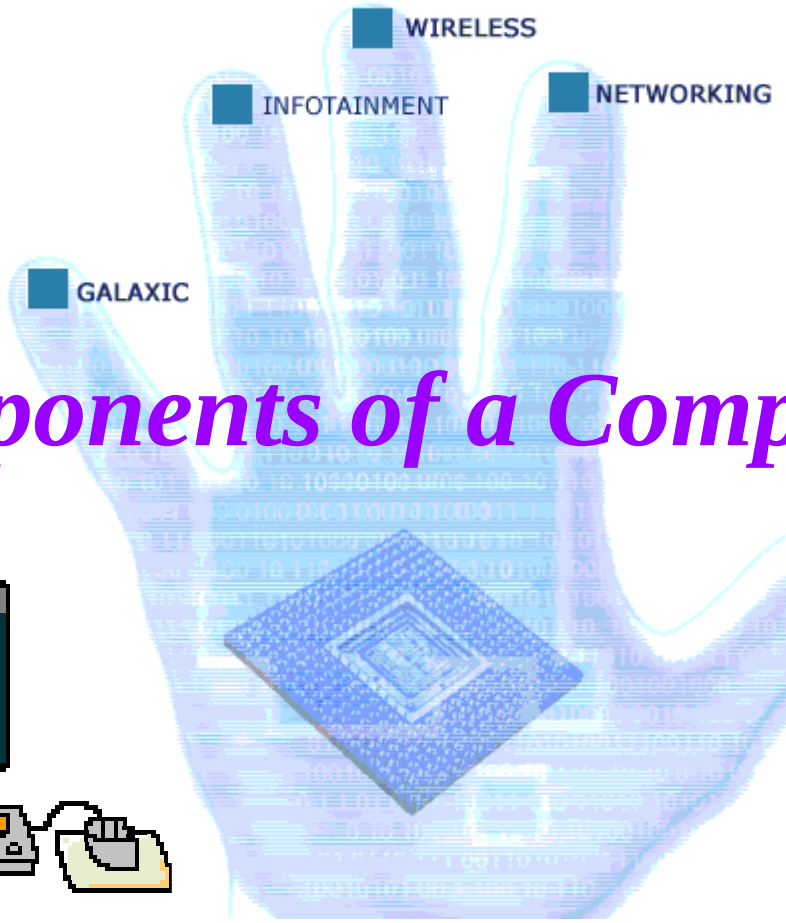


1. Synchronous System Structure



1. Synchronous System Structure

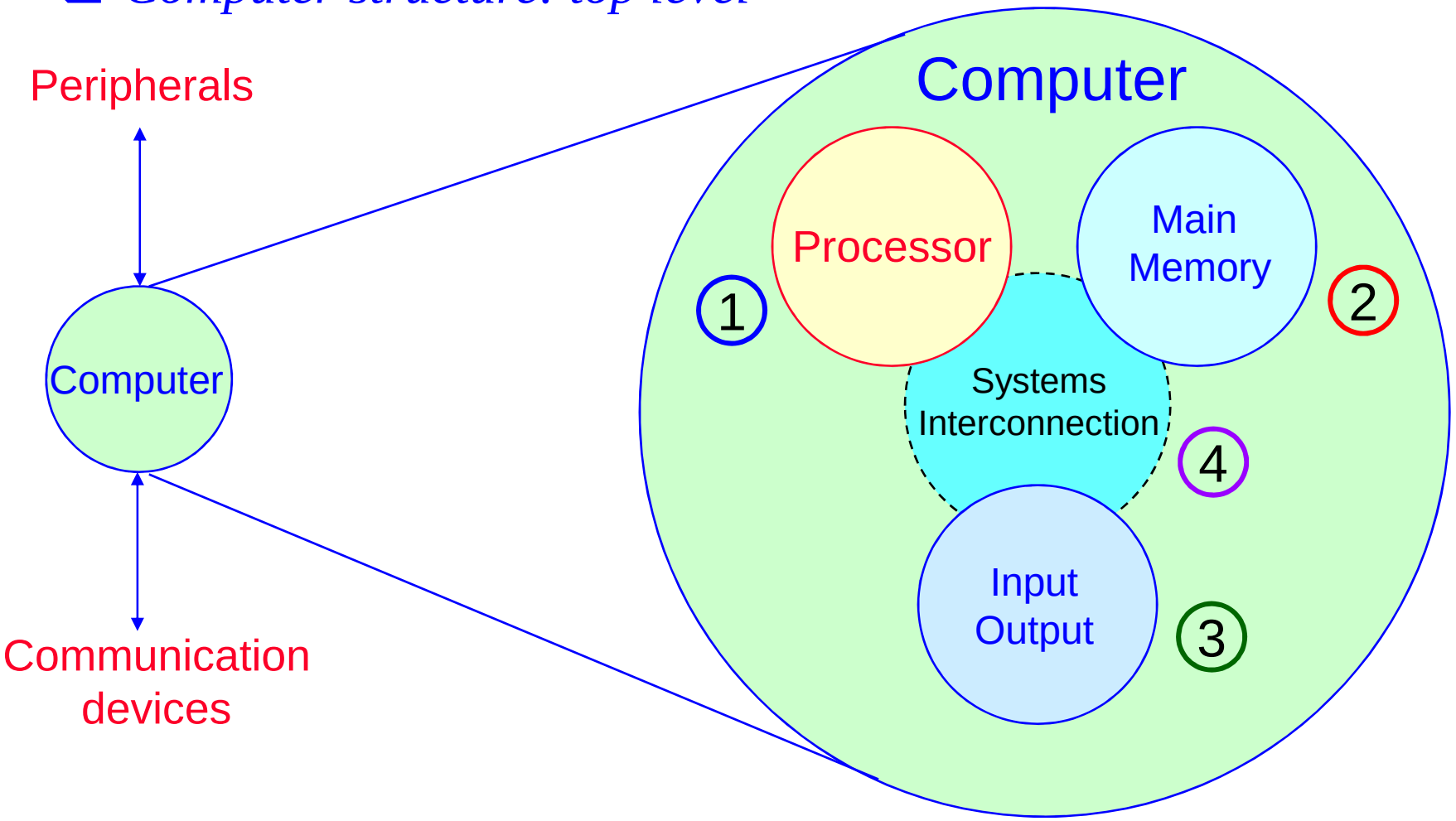




Components of a Computer

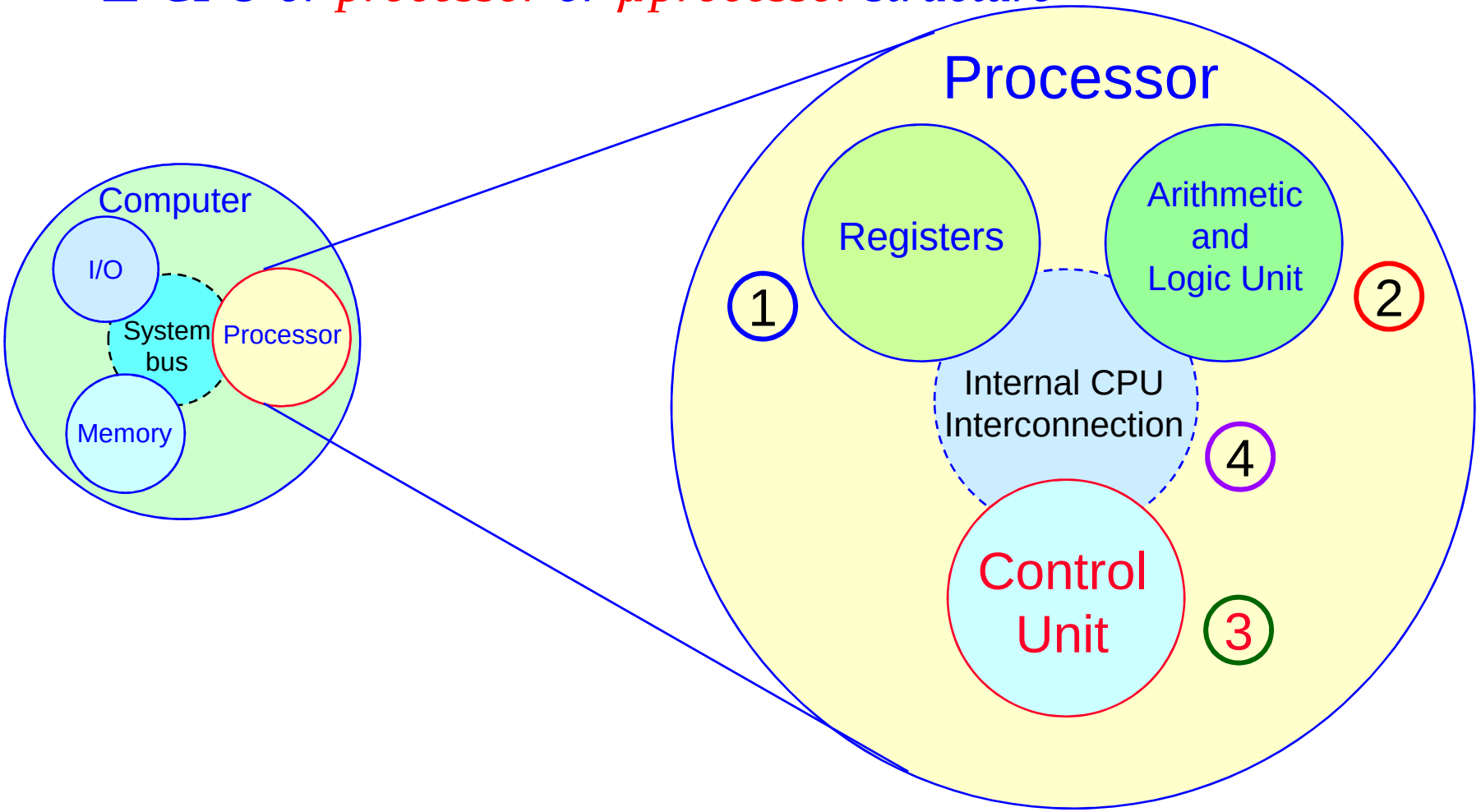
1. Computer: Overview

- *Computer structure: top level*



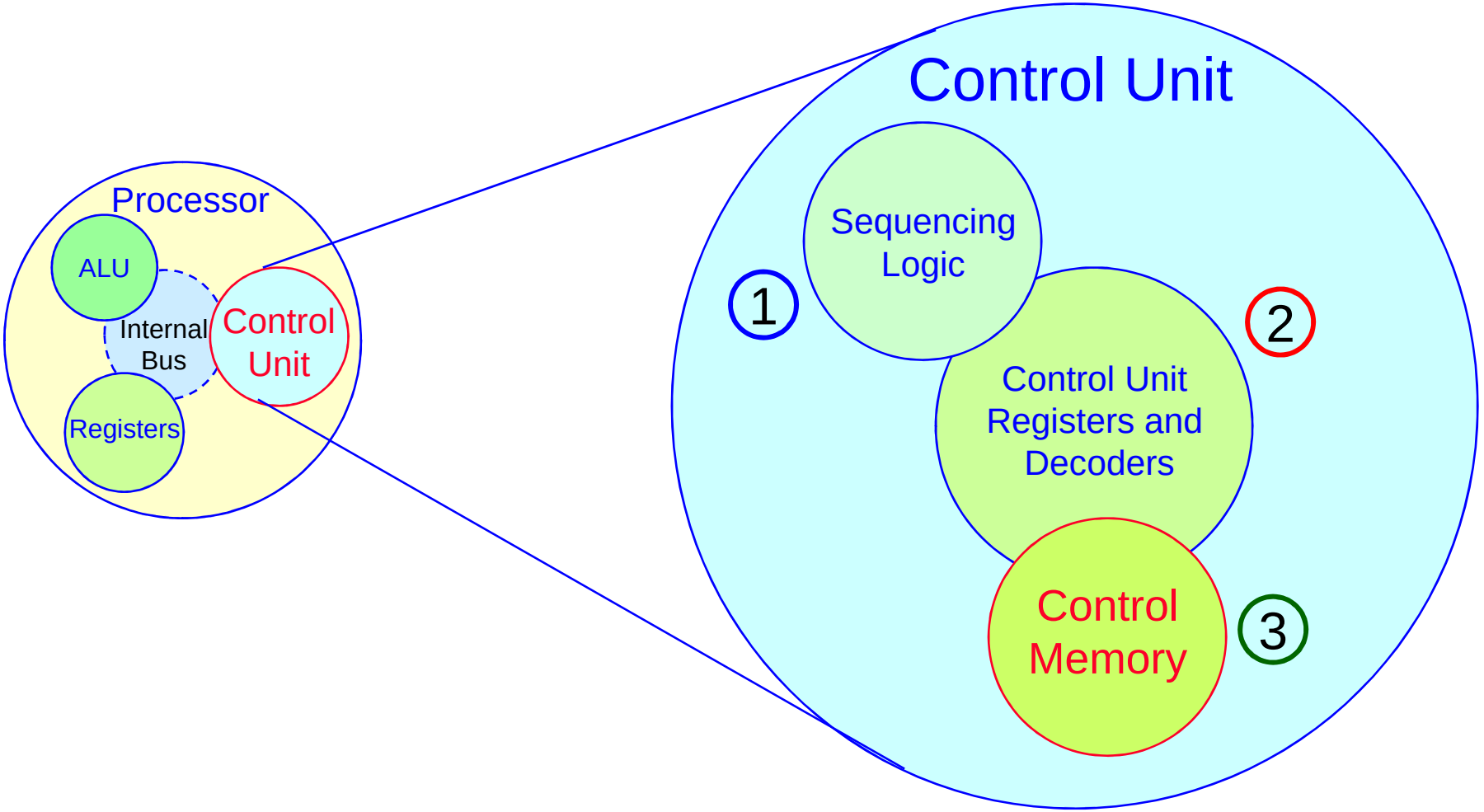
1. Processor: Overview

- *CPU or processor or µprocessor structure*

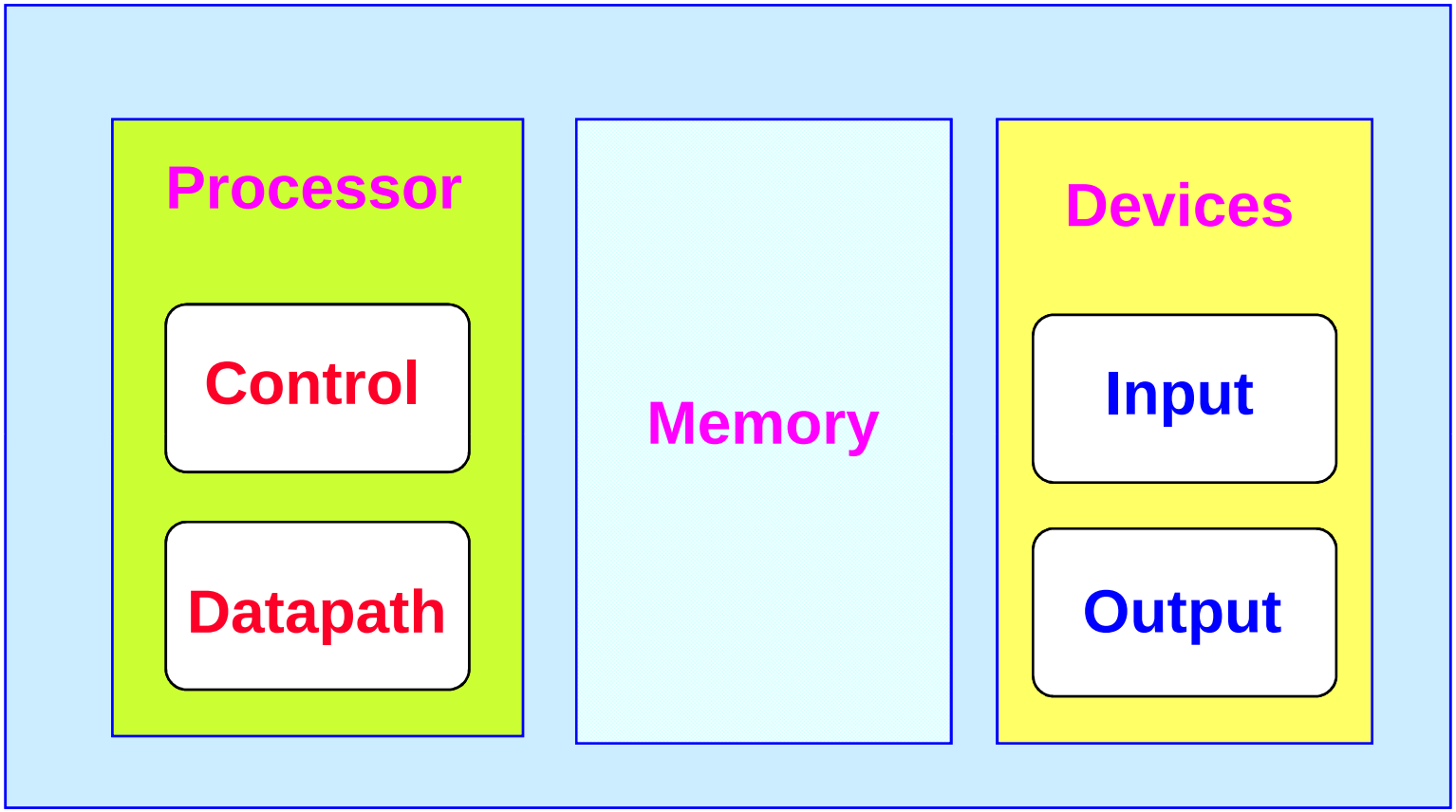


1. Control Unit: Overview

- *Control unit structure*



1. Major Components of a Computer



2. Control Unit

- Control needs to have the
  - Ability to input instructions from memory
  - Logic and means to control instruction sequencing
  - Logic and means to issue signals that control the way information flows between datapath components
  - Logic and means to control what operations the datapath's functional units perform
- Datapath needs to have the
  - Components - functional units (e.g., adder) and storage locations (e.g., register file) - needed to execute instructions
  - Components interconnected so that the instructions can be accomplished
  - Ability to load data from and store data to memory

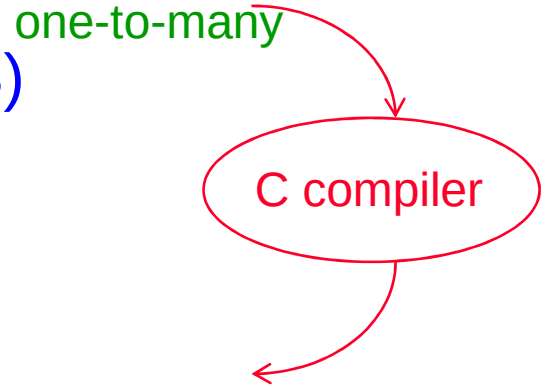
2. Control Unit

- High-level language program (in C)

```
swap (int v[], int k)
```

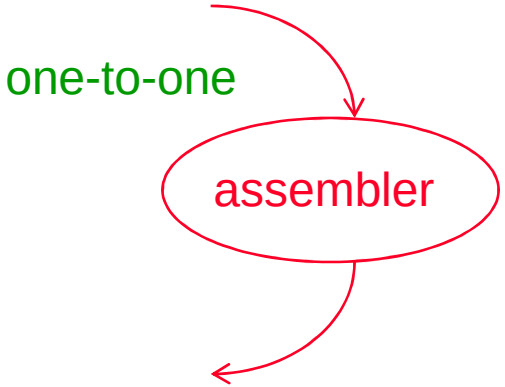
- Assembly language program (for MIPS)

```
swap:   sll    $2, $5, 2
        add    $2, $4, $2
        lw     $15, 0($2)
        lw     $16, 4($2)
        sw     $16, 0($2)
        sw     $15, 4($2)
        jr     $31
```



- Machine (object) code (for MIPS)

```
000000 00000 00101 0001000010000000
000000 00100 00010 0001000000100000
100011 00010 01111 0000000000000000
100011 00010 10000 0000000000000100
101011 00010 10000 0000000000000000
101011 00010 01111 0000000000000100
000000 11111 00000 0000000000001000
```



2. Control Unit

- High-level language program (in C)

```
swap (int v[], int k)
```

- Assembly language program (for MIPS)

```
swap:   sll    $2, $5, 2
        add    $2, $4, $2
        lw     $15, 0($2)
        lw     $16, 4($2)
        sw     $16, 0($2)
        sw     $15, 4($2)
        jr     $31
```

one-to-many

C compiler

- Machine (object) code (for MIPS)

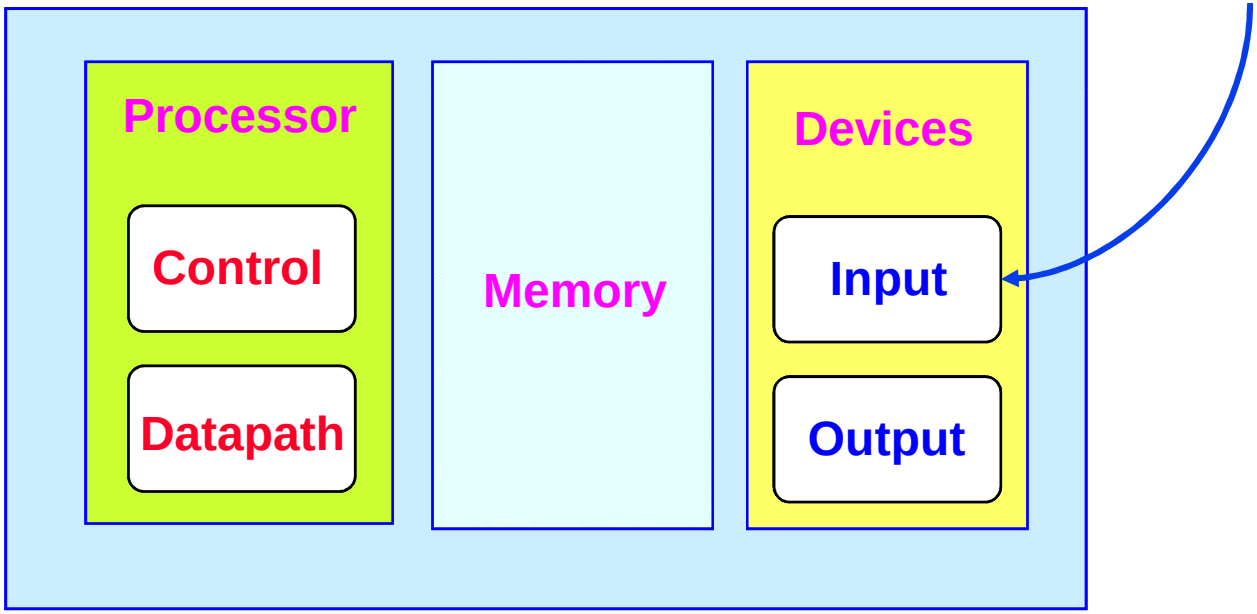
```
000000 00000 00101 00010000010000000
000000 00100 00010 00010000000100000
100011 00010 01111 00000000000000000
100011 00010 10000 00000000000000100
101011 00010 10000 00000000000000000
101011 00010 01111 00000000000000100
000000 11111 00000 00000000000001000
```

one-to-one

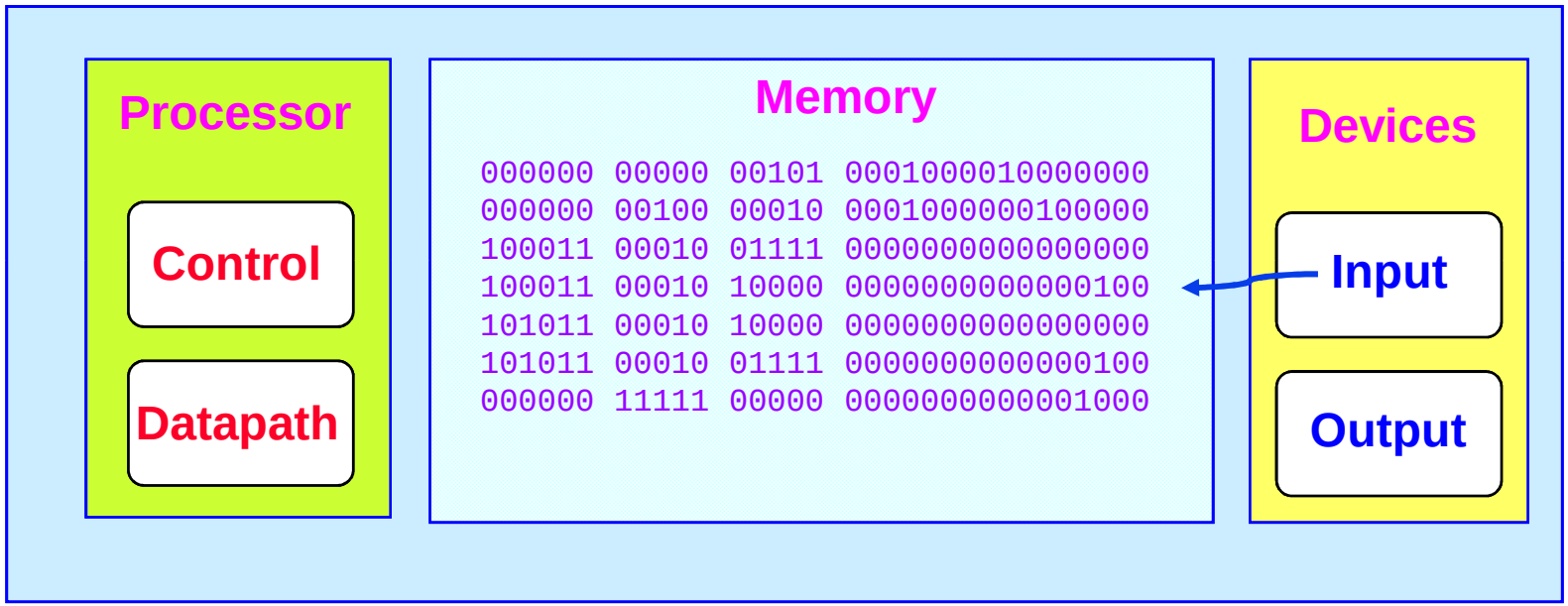
assembler

2. Input Device Inputs Object Code

```
000000 00000 00101 0001000010000000
000000 00100 00010 0001000000100000
100011 00010 01111 0000000000000000
100011 00010 10000 0000000000000100
101011 00010 10000 0000000000000000
101011 00010 01111 0000000000000100
000000 11111 00000 0000000000001000
```

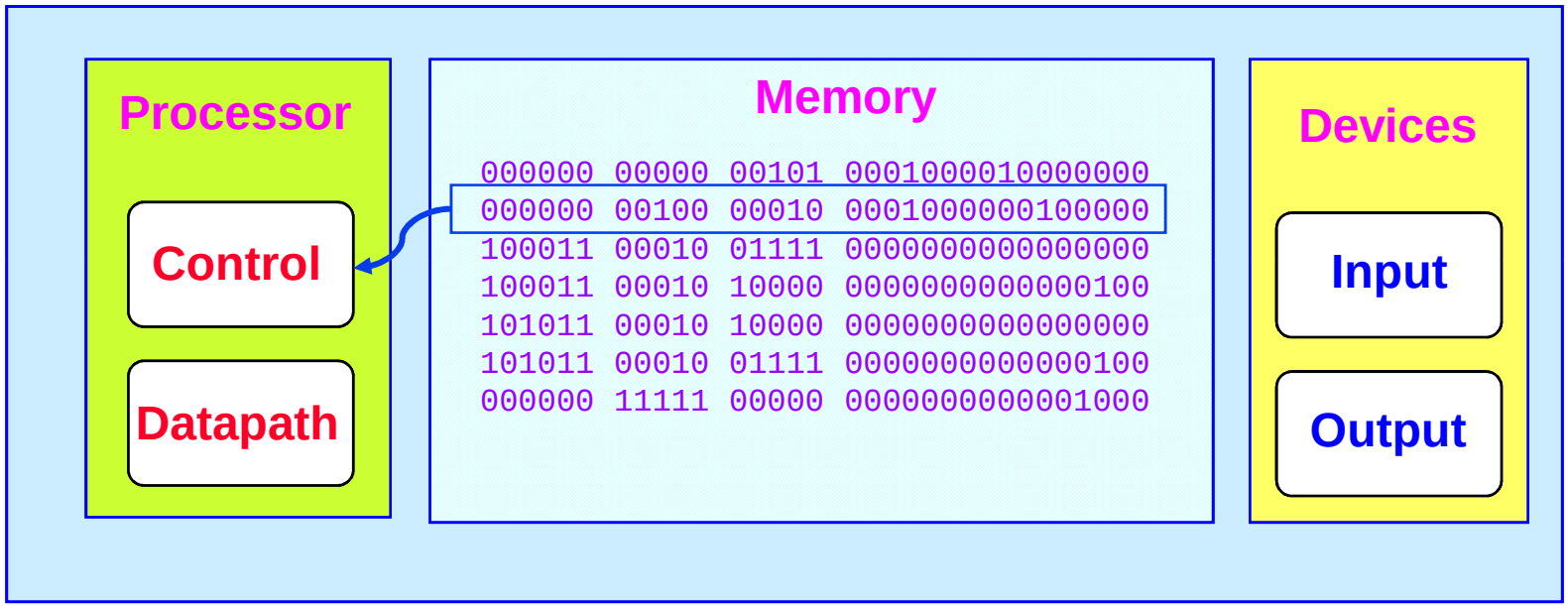


2. Object Code Stored in Memory



2. Processor Fetches an Instruction

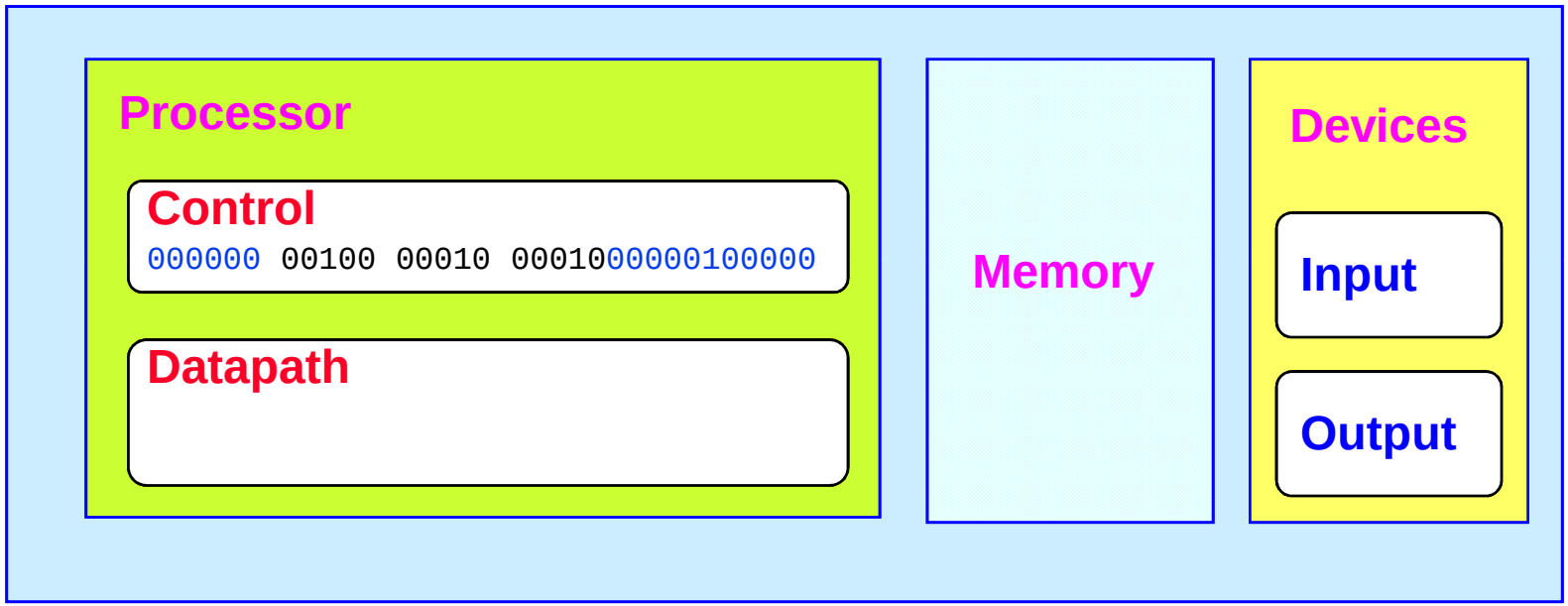
Processor fetches an instruction from memory



Where does it fetch from?

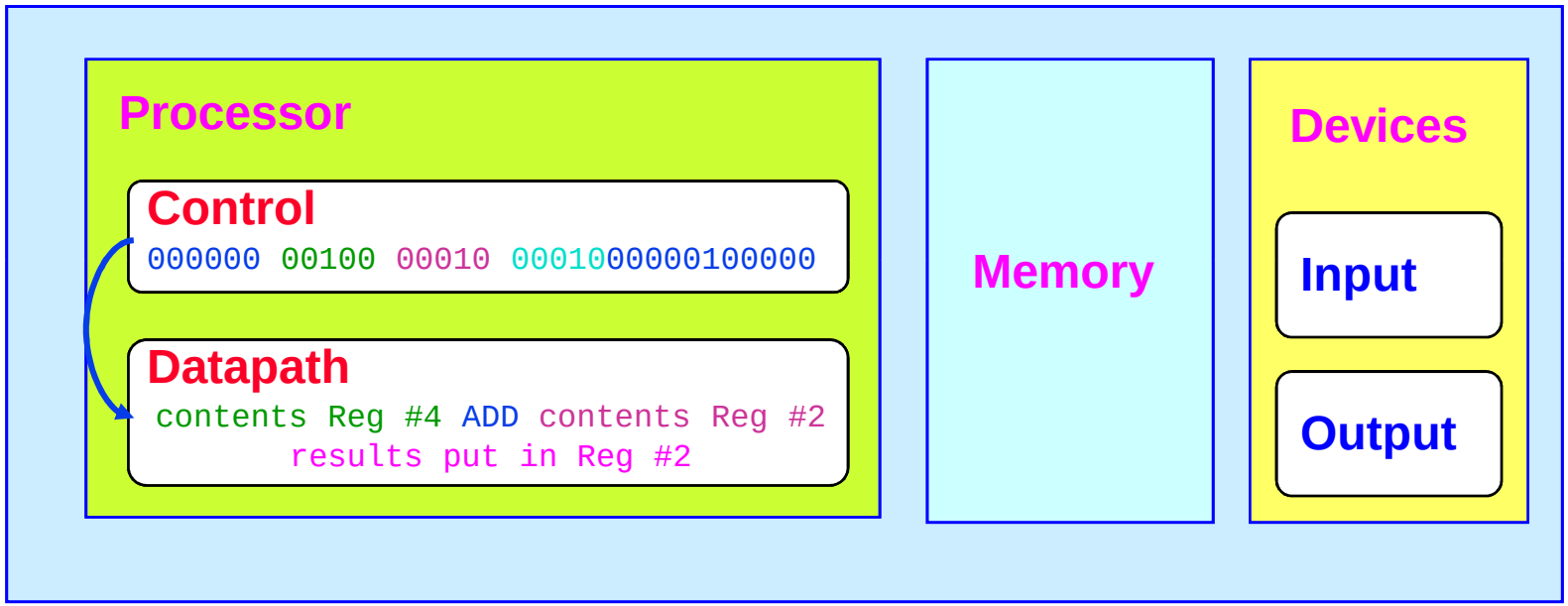
2. Control Decodes the Instruction

Control decodes the instruction to determine what to execute

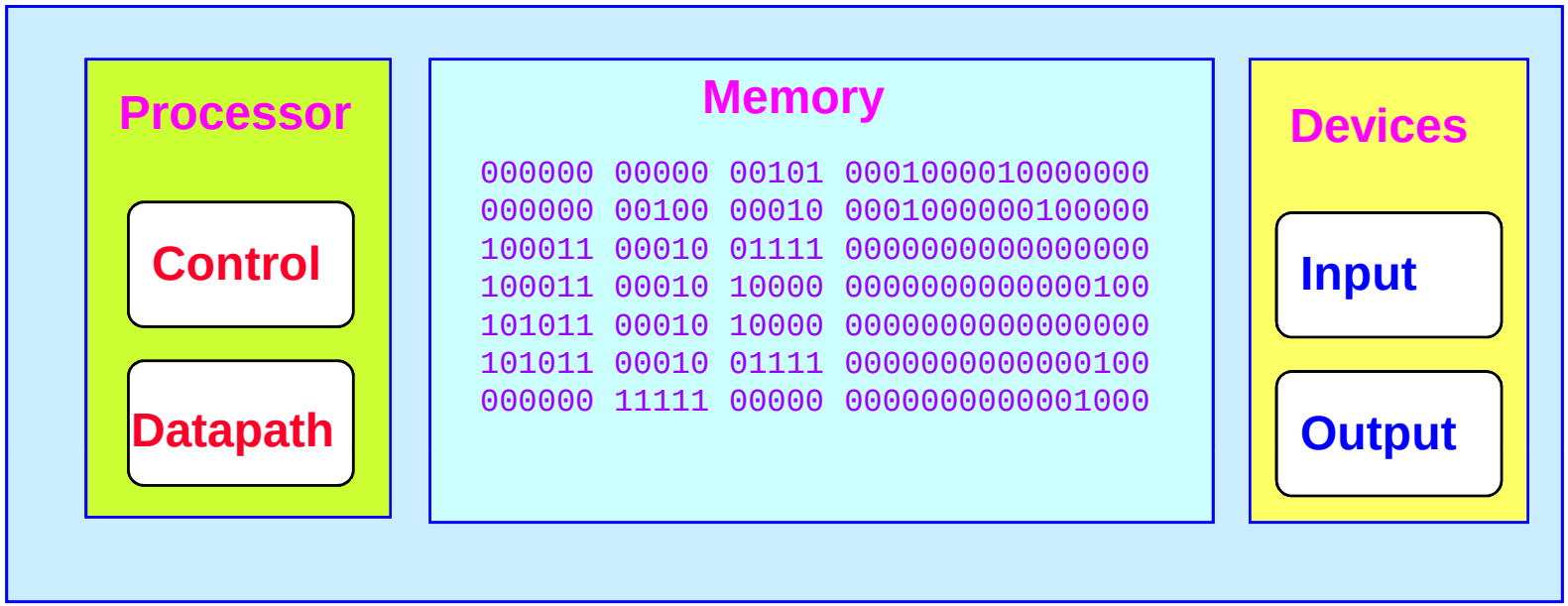


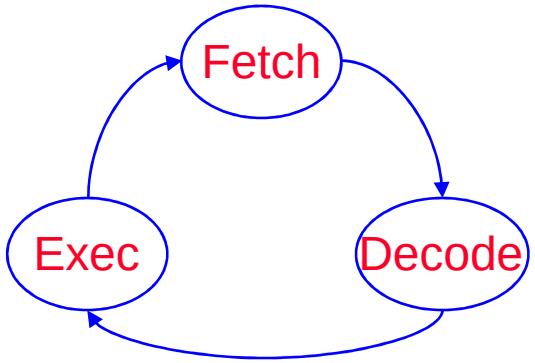
2. Datapath Executes the Instruction

Datapath executes the instruction as directed by control

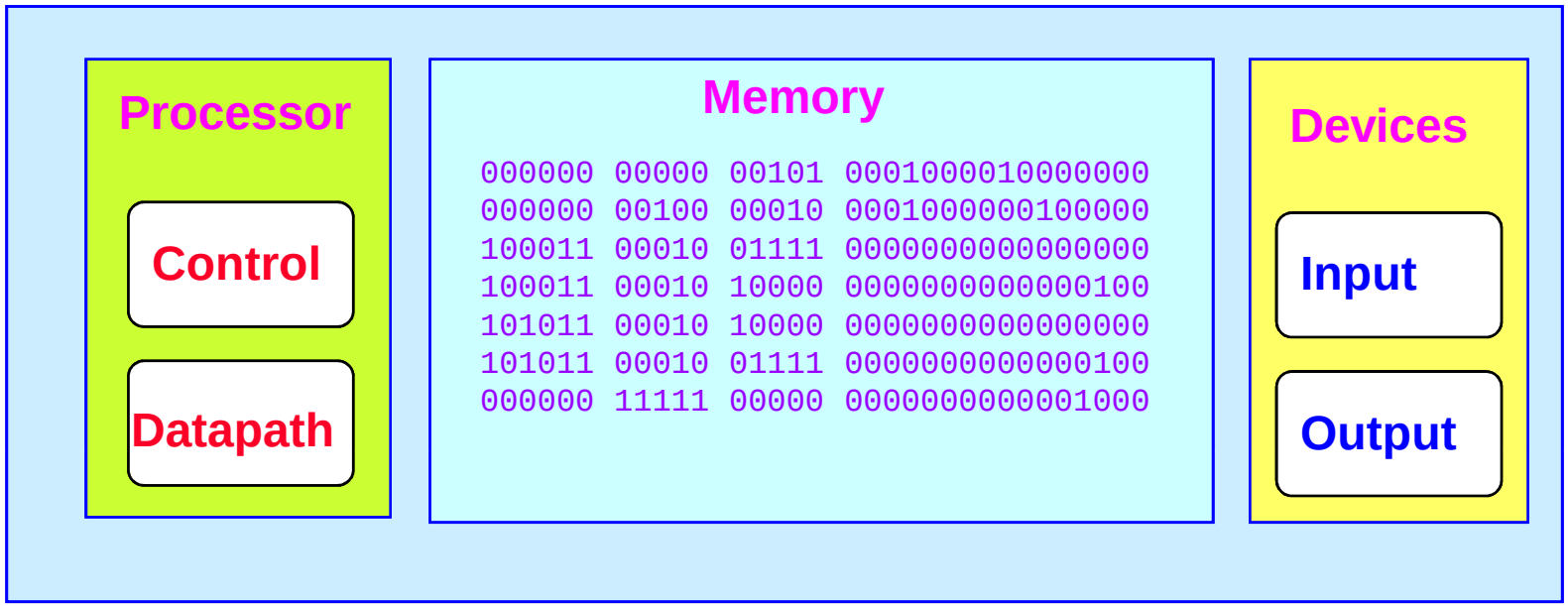


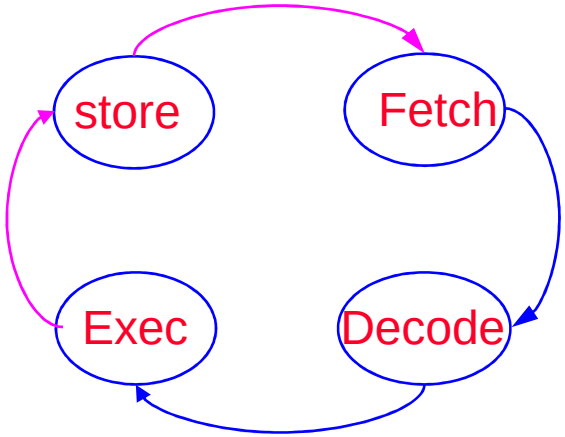
2. What Happens Next?





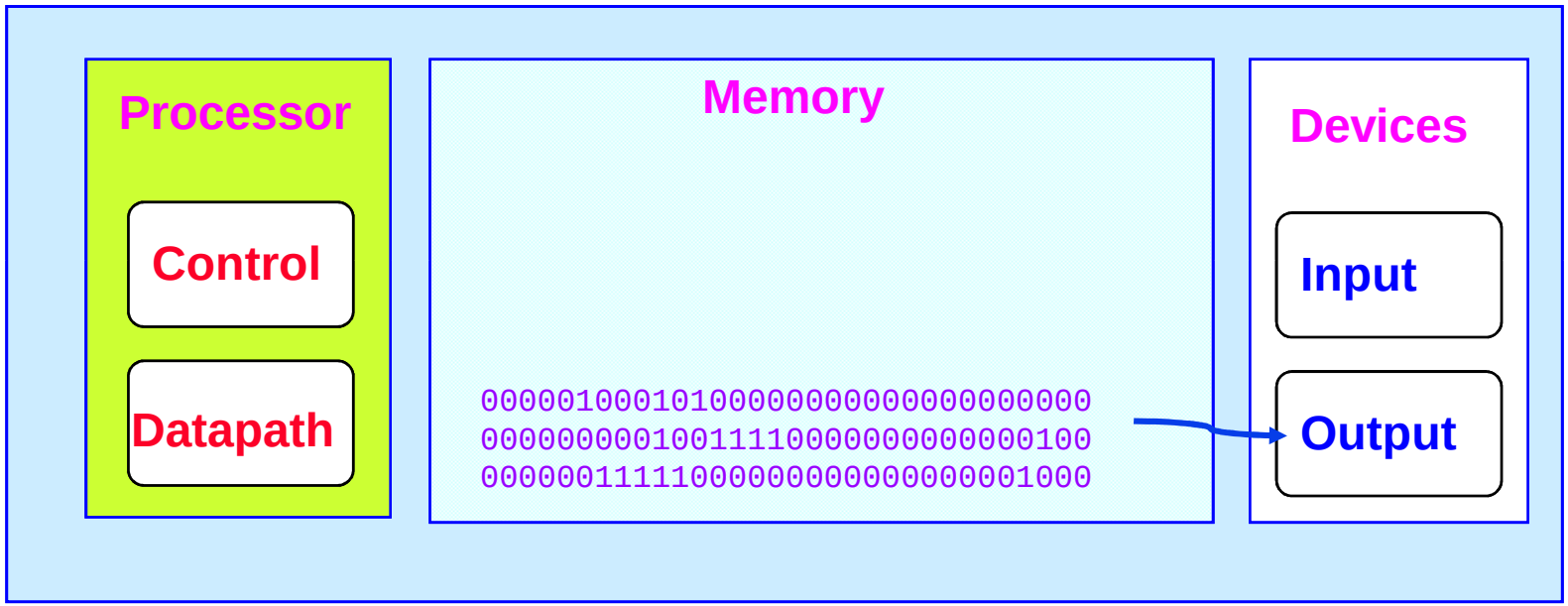
2. What Happens Next?



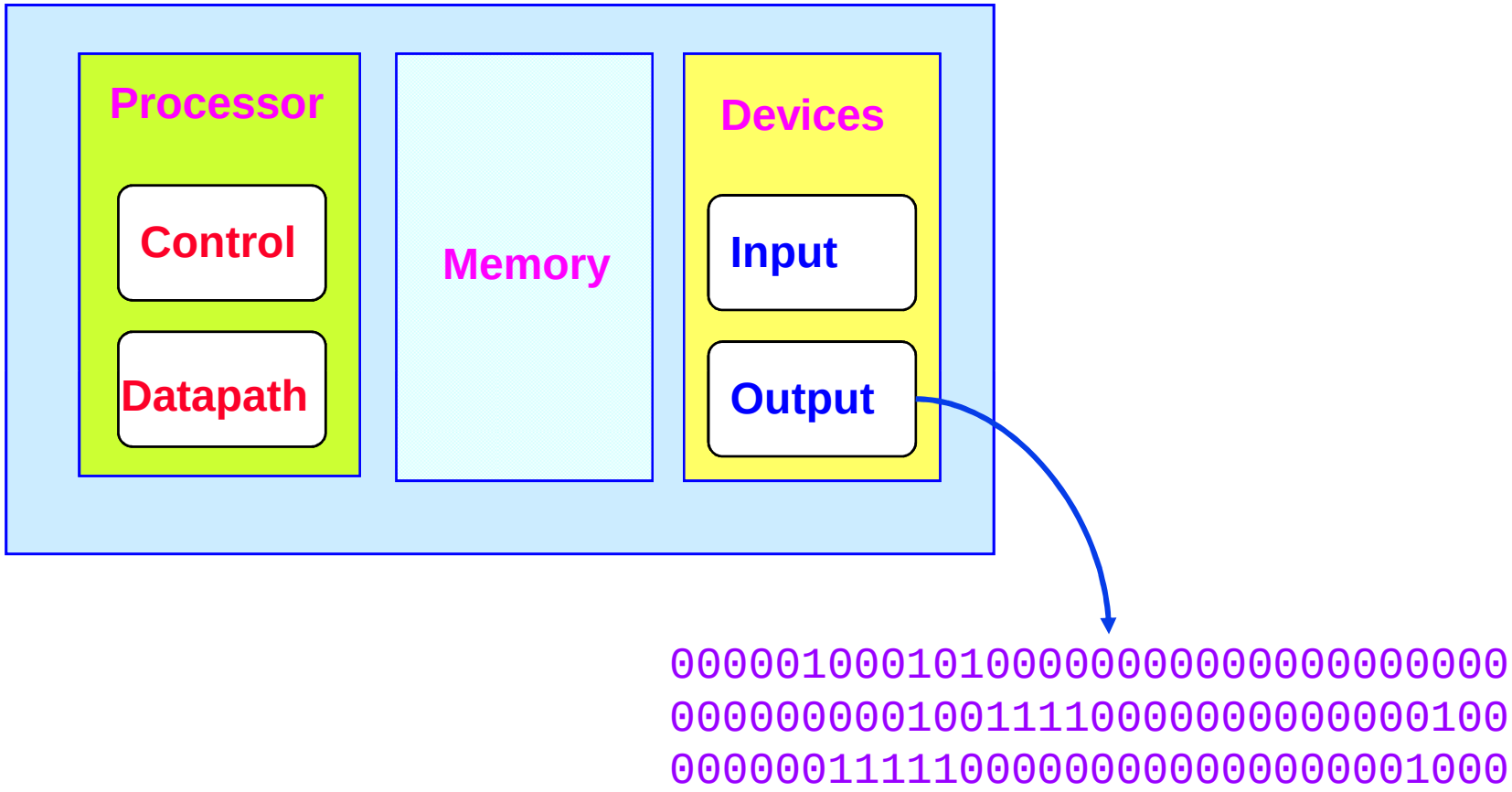


2. Output Data Stored in Memory

At program completion the data to be output resides in memory



2. Output Device Outputs Data



Design of Control Units

1. Control Unit: Basic Concepts

- *Control unit* is the part of a *processor (CPU)* or other device that *directs its operation*
- In *modern computers or processors*, each subsystem may have its own *subsidiary controller*, with the *control unit* acting as a *supervisor*
- *Control unit* can be thought of as a *finite state machine*
- *Finite-state machine (FSM)* or *finite-state automaton (FSA)* is an *abstract machine* that has *only a finite*, and constant amount of *memory*

1. Control Unit: Basic Concepts

- There are *several types* of *finite state machines*:
  - *acceptors* produce a *"yes"* or *"no" answer* to the input; they either accept the input or do not
  - *recognizers* categorize the input
  - *transducers* are used to *generate an output* from a *given input*
- *Finite state machines* are implemented in *hardware*, where *the input*, *the state* and *the output* are *bit vectors* of *fixed size (Moore machines and Mealy machines)*
- *Mealy machines* have actions *(outputs)* associated with *transitions* and *Moore machines* have actions associated with *states*

1. Control Unit: Basic Concepts

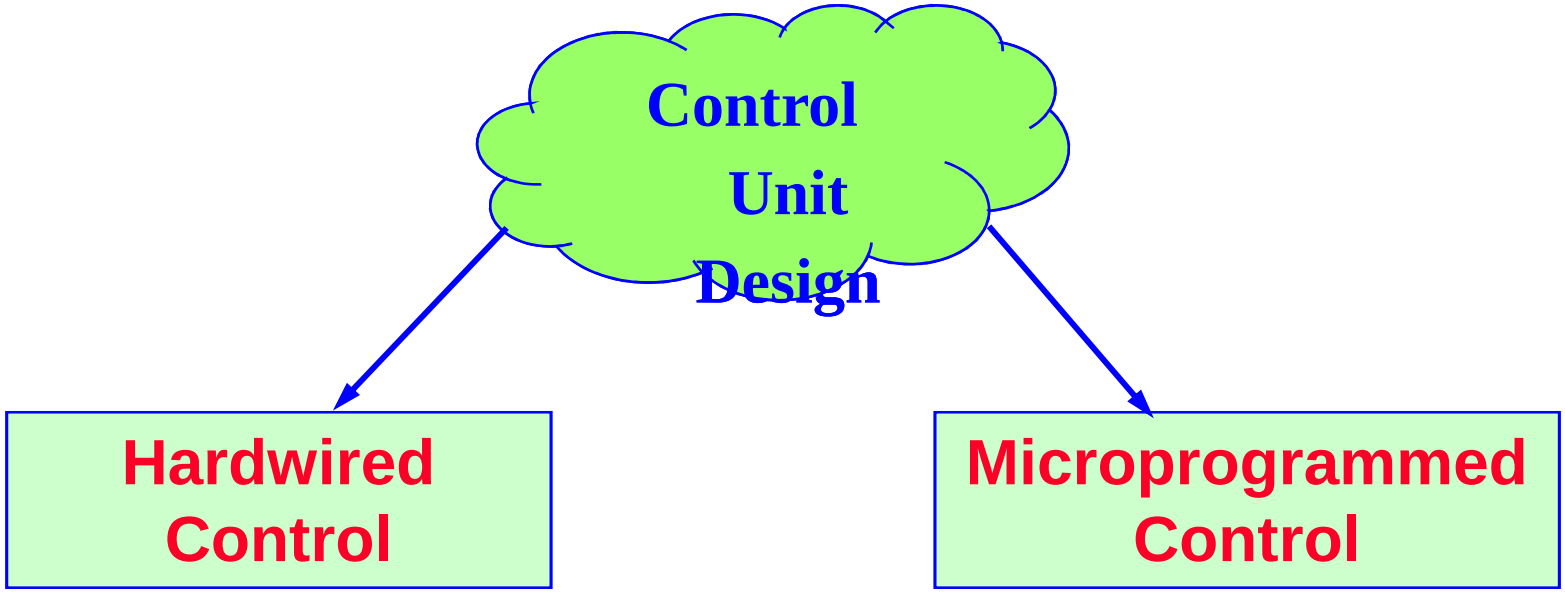
- In *deterministic automata*, for *each state* there is at *most one transition for each possible input*
- In *non-deterministic automata*, there can be *more than one transition* from a given state for a *given possible input*
- *Non-deterministic automata* are usually implemented by converting them to deterministic automata

2. Control Unit Implementations

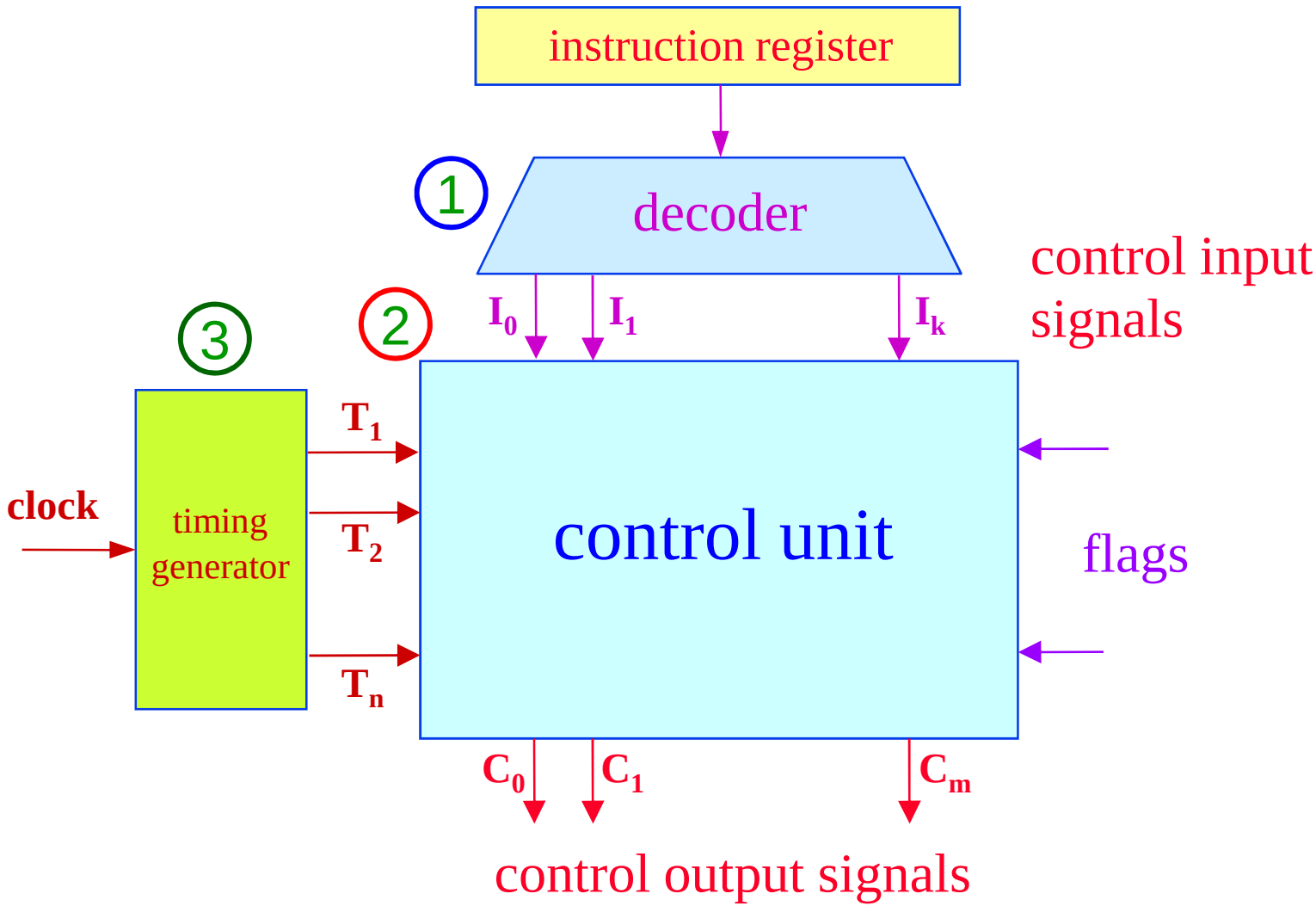
- There are two types of control unit implementations:
  - *hardwired implementation*
  - *micro-programmed implementation*
- *Hardwired* implementation
  - combinational circuit
  - *input signals* are *transformed* into a *set of output signals*
- *Micro-programmed* implementation
  - *microprogram* stored in a *control store (memory)*

2. Control Unit Implementations

- *Implementation types*:



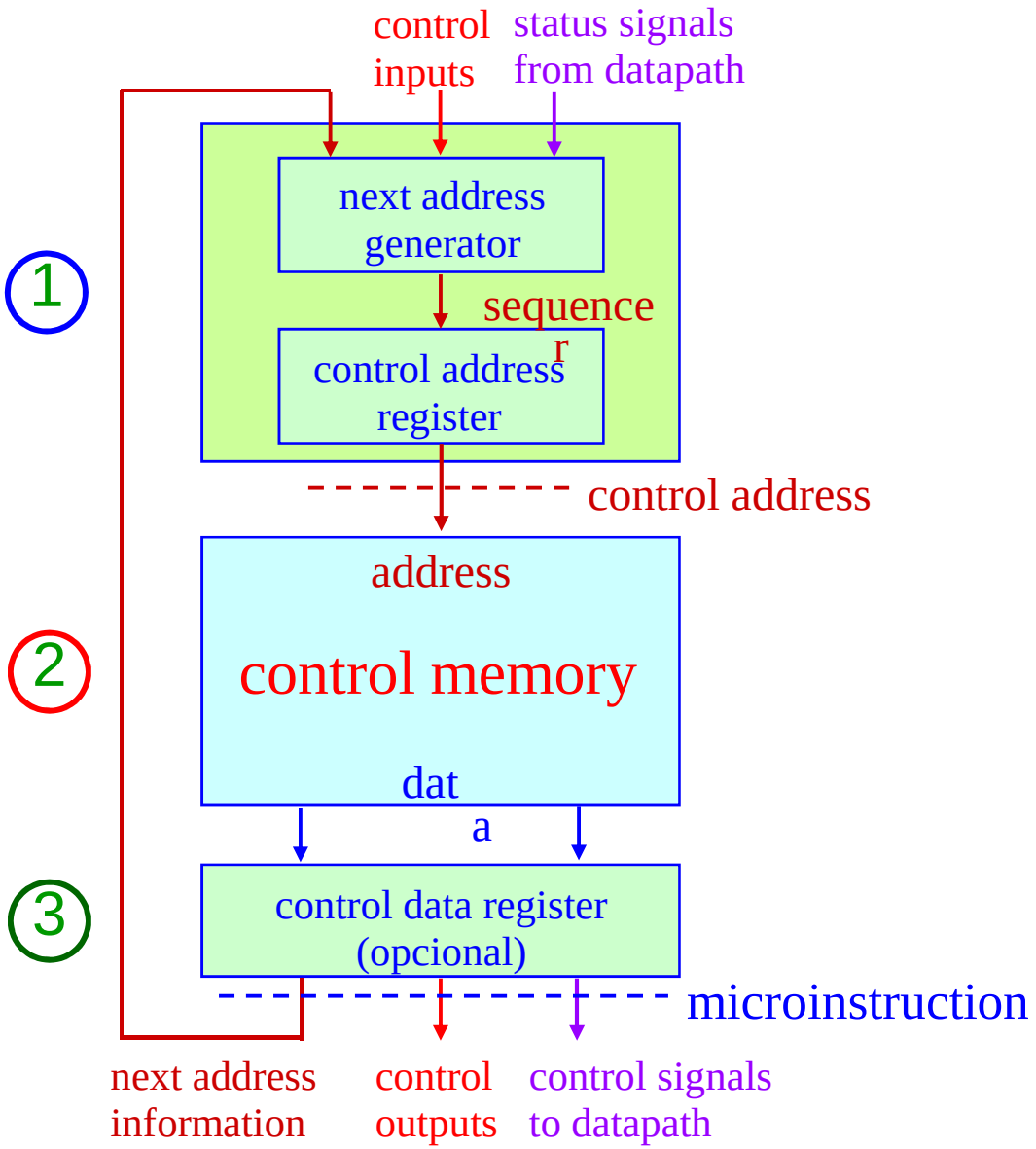
2.1 Hardwired Control Unit



2.1 Hardwired Control Unit

- *Control unit: combinational circuit*
- *Control unit inputs* can be:
  - *flags*
  - *control signals from the instruction register*
- *Control unit inputs* need to be *decoded*
  - decoder takes *n bits input* from the *instruction register*, and *generates 2n bits output* (*n is opcode length*)
- *Timing generator*
  - issues a *repetitive sequence of pulses*
  - counter *generates T1, T2…* for the time units

2.2 Micro programmed Control Unit



2.2 Micro programmed Control Unit

- *Micro-programmed control units* are *implemented* as a *microprogram* which is *stored* in a *control store memory*
- *Microprogram* is a *program* which consists of a *microcode* that *controls* the different parts of a processor.
- *The memory* in which the microprogram *resides* is called a *control store memory*
- *Words of the microprogram* are usually *accessed* or *selected* by a *microsequencer* or *sequencer,* which generates *the addresses* for the *memory*

2.2 Microprogrammed Control Unit

- *The bits from words of the microprogram* directly *control* the different parts of the processor or device
- *Address are generated* by some *combination* of a *counter*, a *field from a microinstruction*, and some *subset of the instruction register*
- *Counter* is used for the *typical case*, which generates the address of the next microinstruction
- The *simplest sequencer* is just a *register* loaded from a *few bits* of the *control store*

2.2 Microprogrammed Control Unit

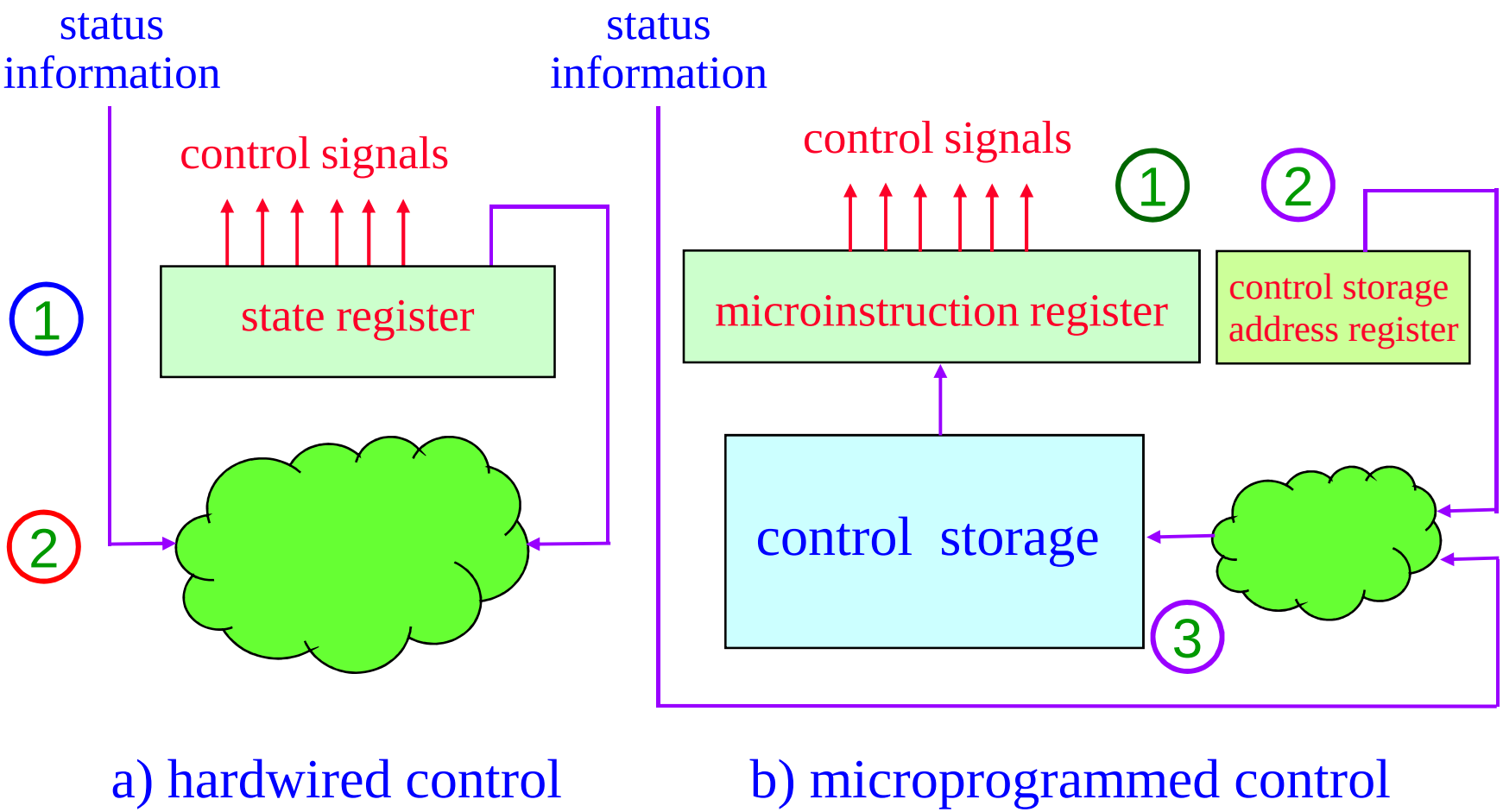
- *Example microinstruction: typically in ROM-EEPROM*

	Nxt1	Nxt0	Cn1	Cn2	Cn3	Cn4	Cn5	Cn6
A	B	A	0	0	0	0	0	1
B	C	D	1	0	0	1	0	0
C	D	D	0	0	1	0	0	0
D	E	E	1	0	1	0	0	0
E	F	F	0	0	0	0	1	0
F	A	A	0	1	1	0	0	0

Each Cn bit is a control line

Nxt is next address, depending on decision

2.3 Hardwired vs Microprogrammed Control



a) hardwired control

b) microprogrammed control

2.3 Hardwired vs Microprogrammed Control

- *There is no intrinsic difference*:
  - the pair of *"microinstruction-register"* and *"control storage address register"* can be regarded as a *"state register"* for the *hardwired control*
  - the *control storage* can be regarded as a *combinational logic circuit*. We can *assign any 0, 1 values to each output corresponding to each address*, *which can be regarded as the input for a combinational logic circuit*. This is *a truth table*
  - the same *field configuration (state assignment)* can be used for both of these two types of control
  - *any kind of sophisticated control* can be implemented by using these two types of control, however the *hardwired control have been historically faster*

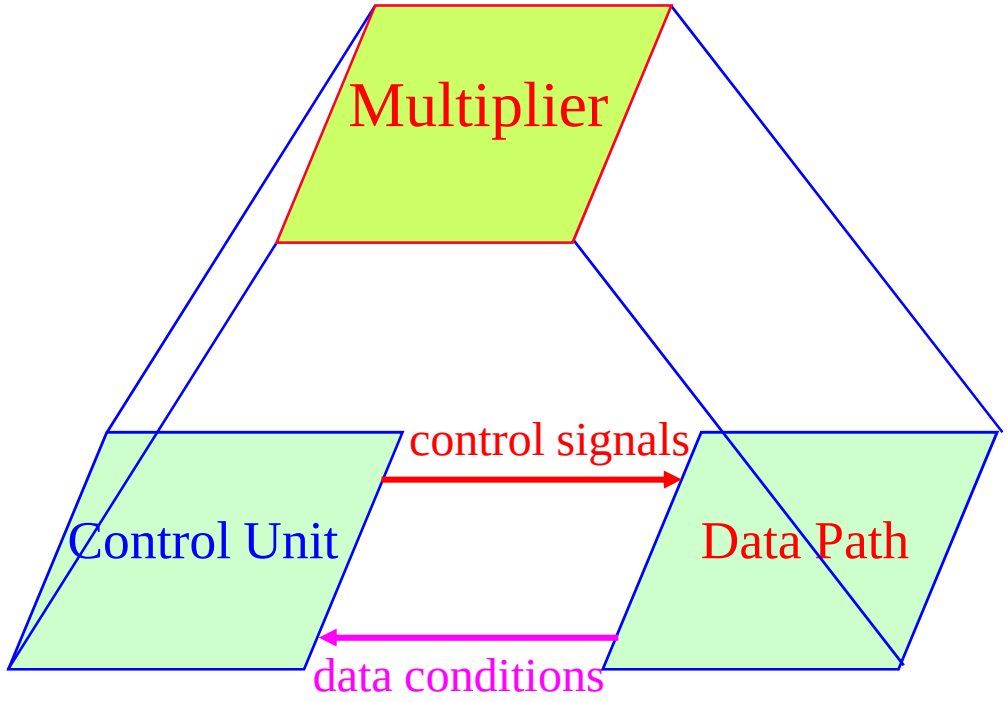
2.3 Hardwired vs Microprogrammed Control

- *There is no intrinsic difference*:
  - *microprogrammed control* is not always necessary to implement *CISC processors*
  - *CISC processors* also can be implemented by using *hardwired control*
  - *CISC and RISC processors* are the major two different types of *ordinary SISD machines*

Hardwired Control

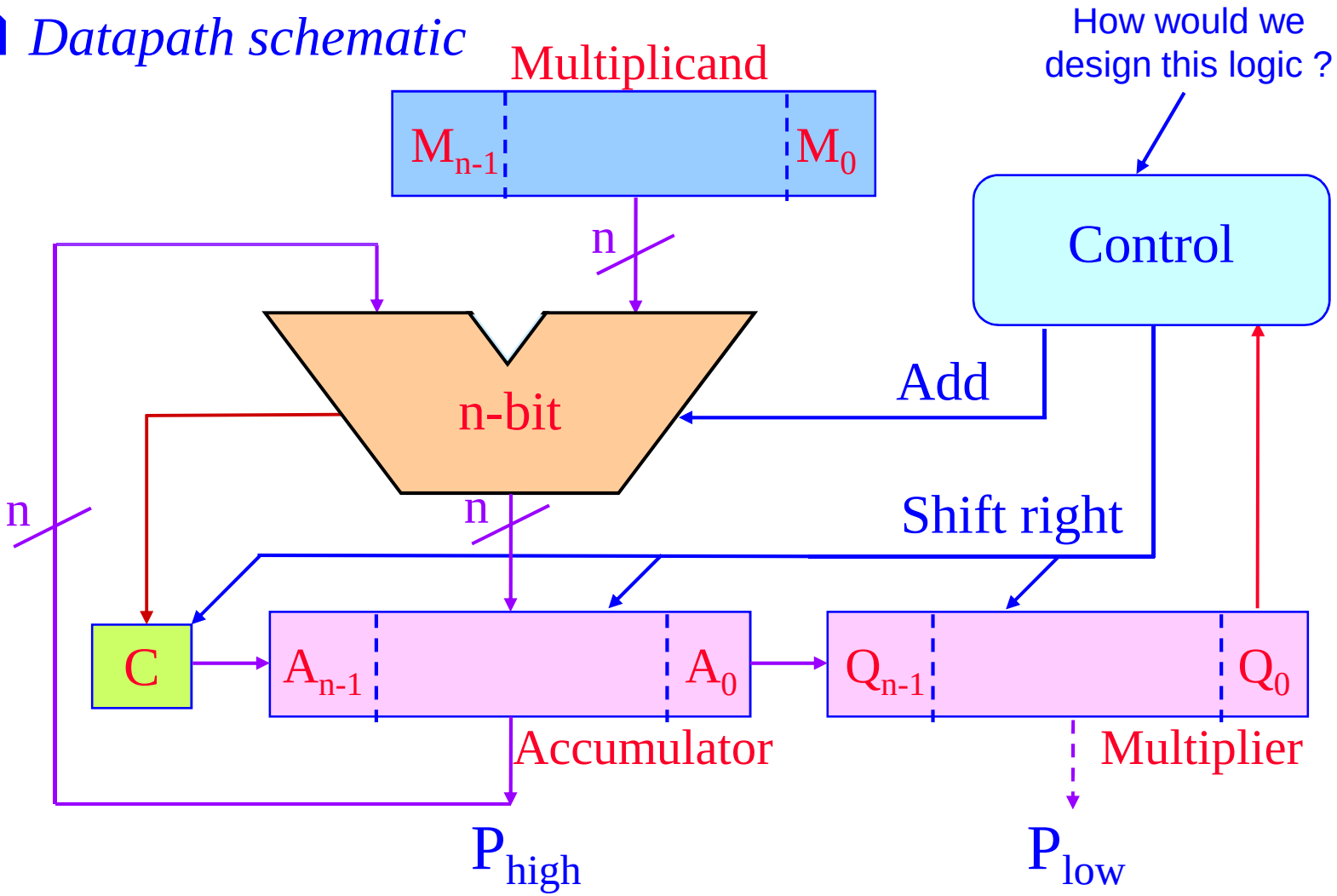
1. Hardwired Control: Binary Multiplier

- *Binary multiplier block diagram*



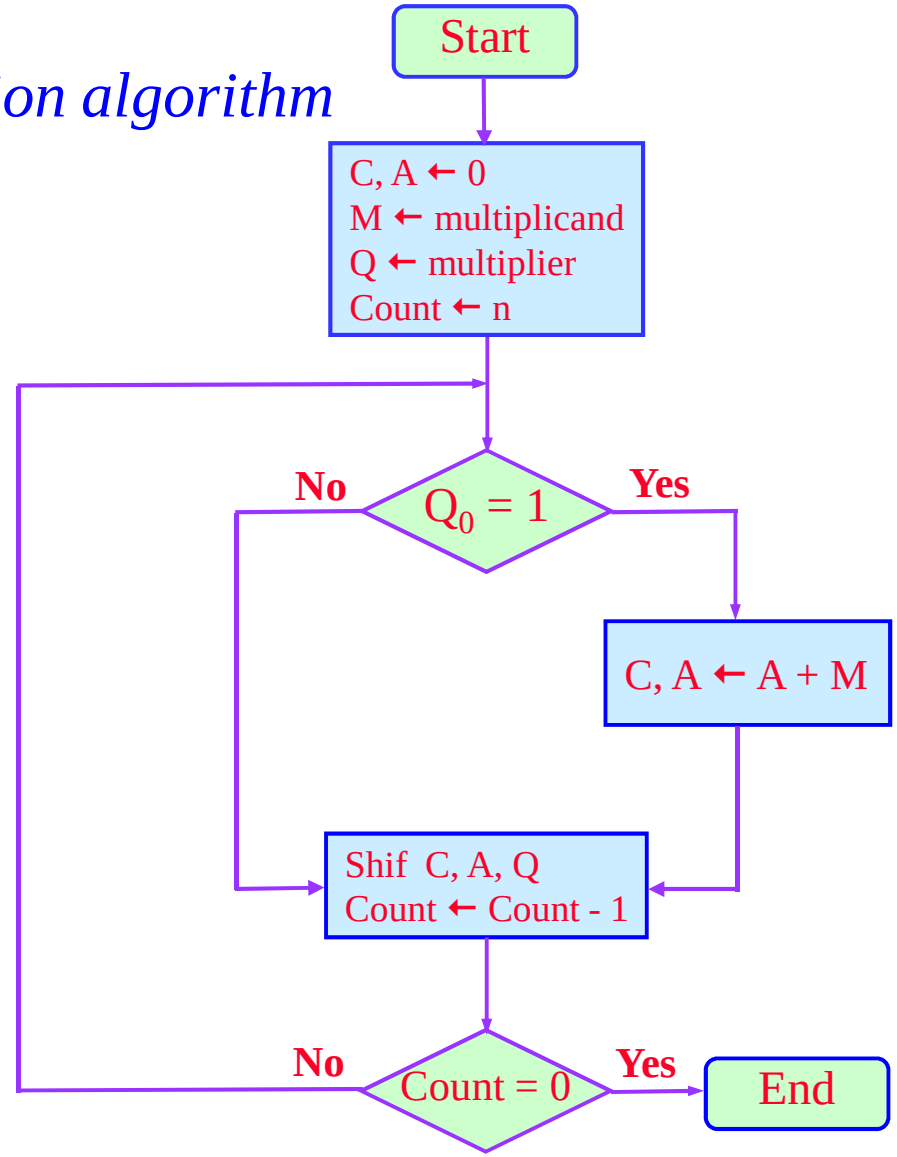
1. Hardwired Control: Binary Multiplier

- *Datapath schematic*



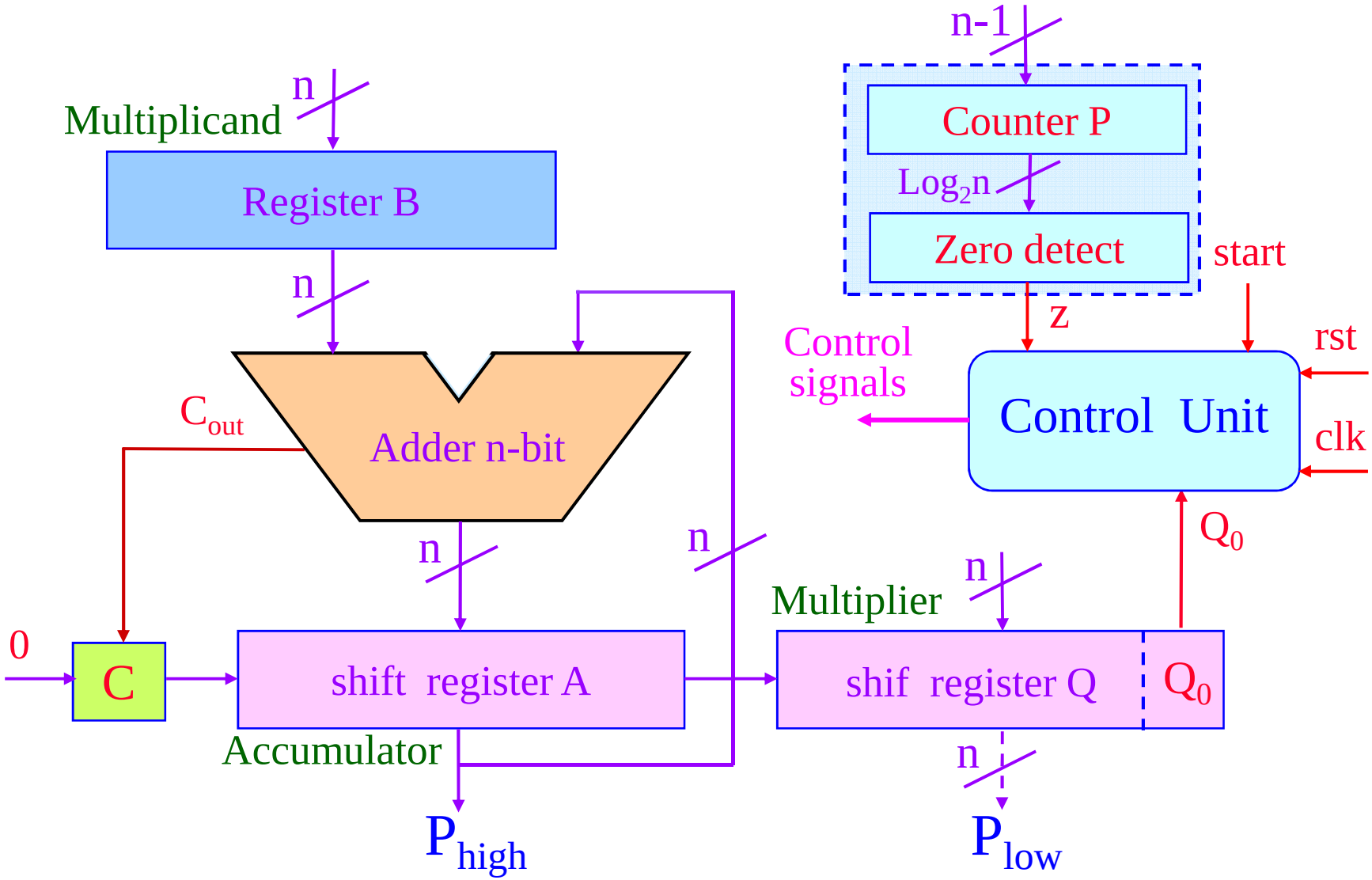
1. Hardwired Control: Binary Multiplier

- *Multiplication algorithm*



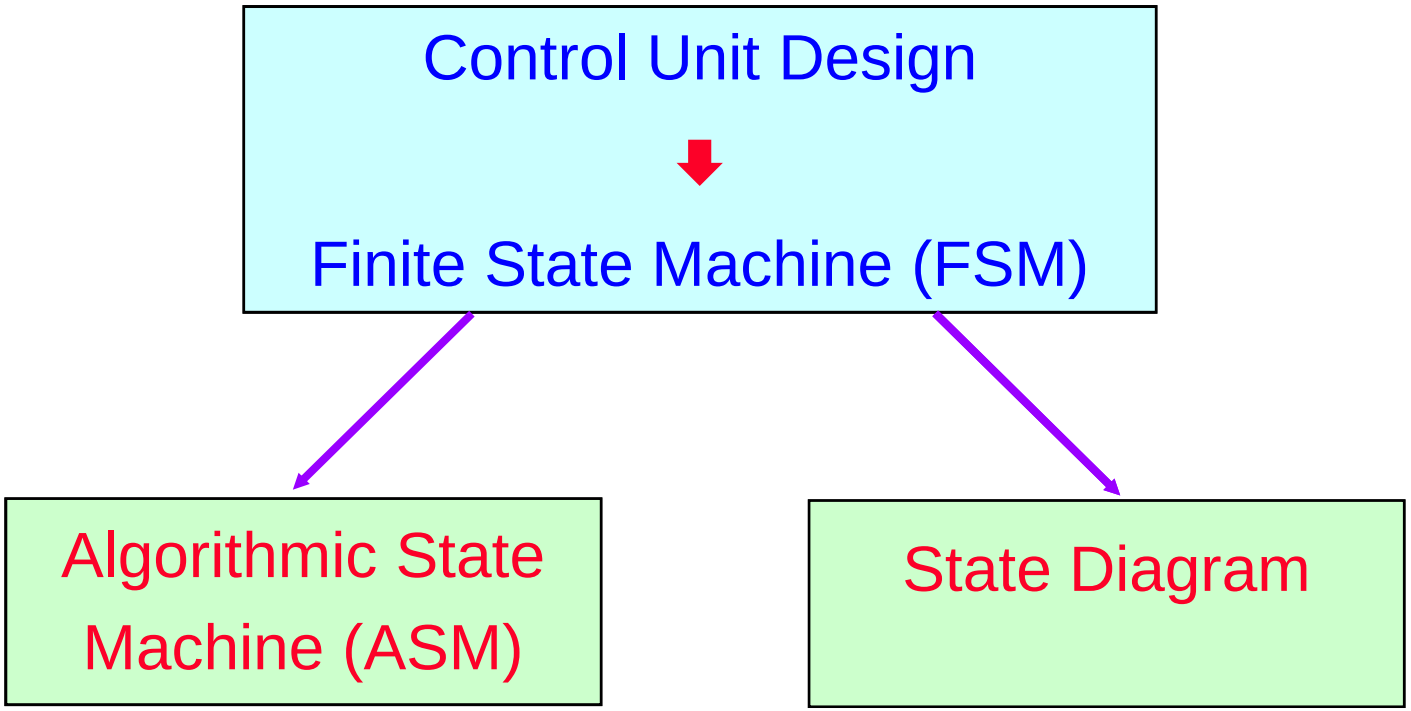
Product in A, Q

1. Hardwired Control: Binary Multiplier



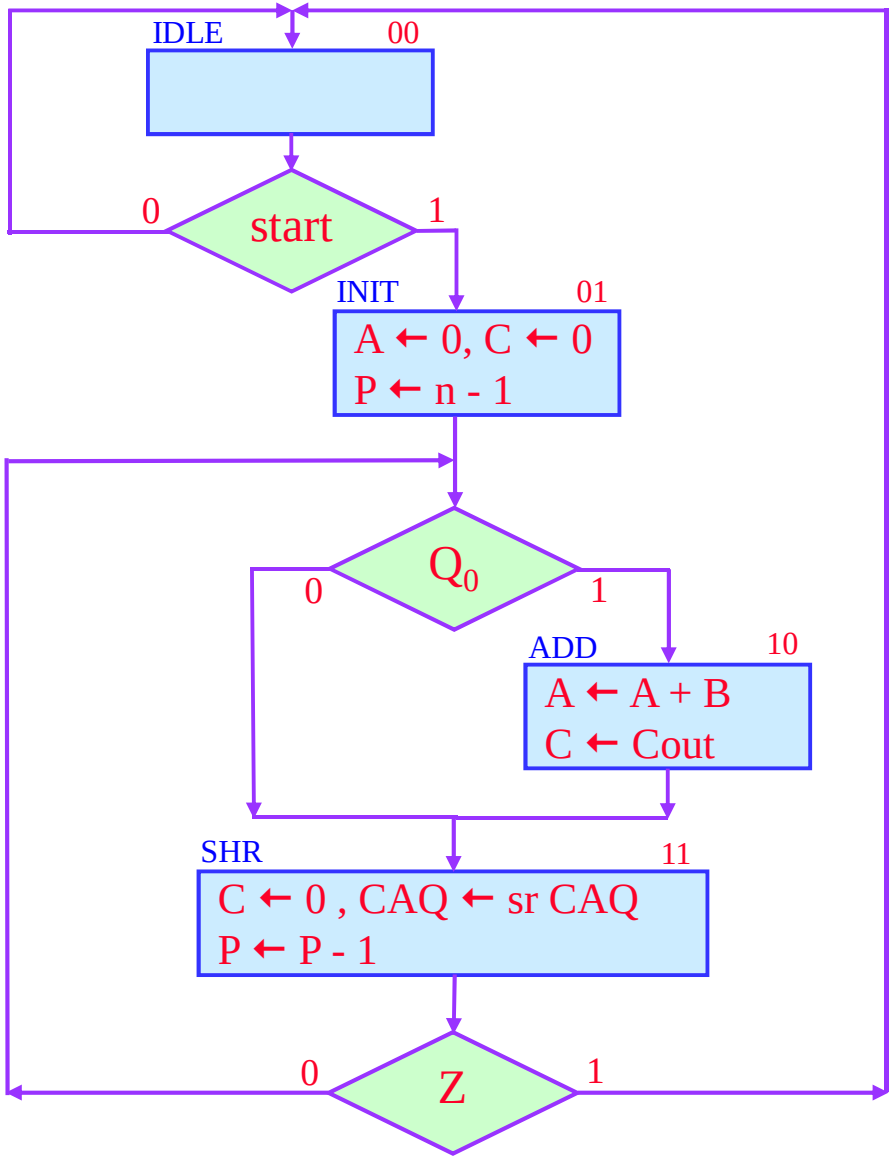
1. Hardwired Control: Binary Multiplier

- *Description of finite state machine*



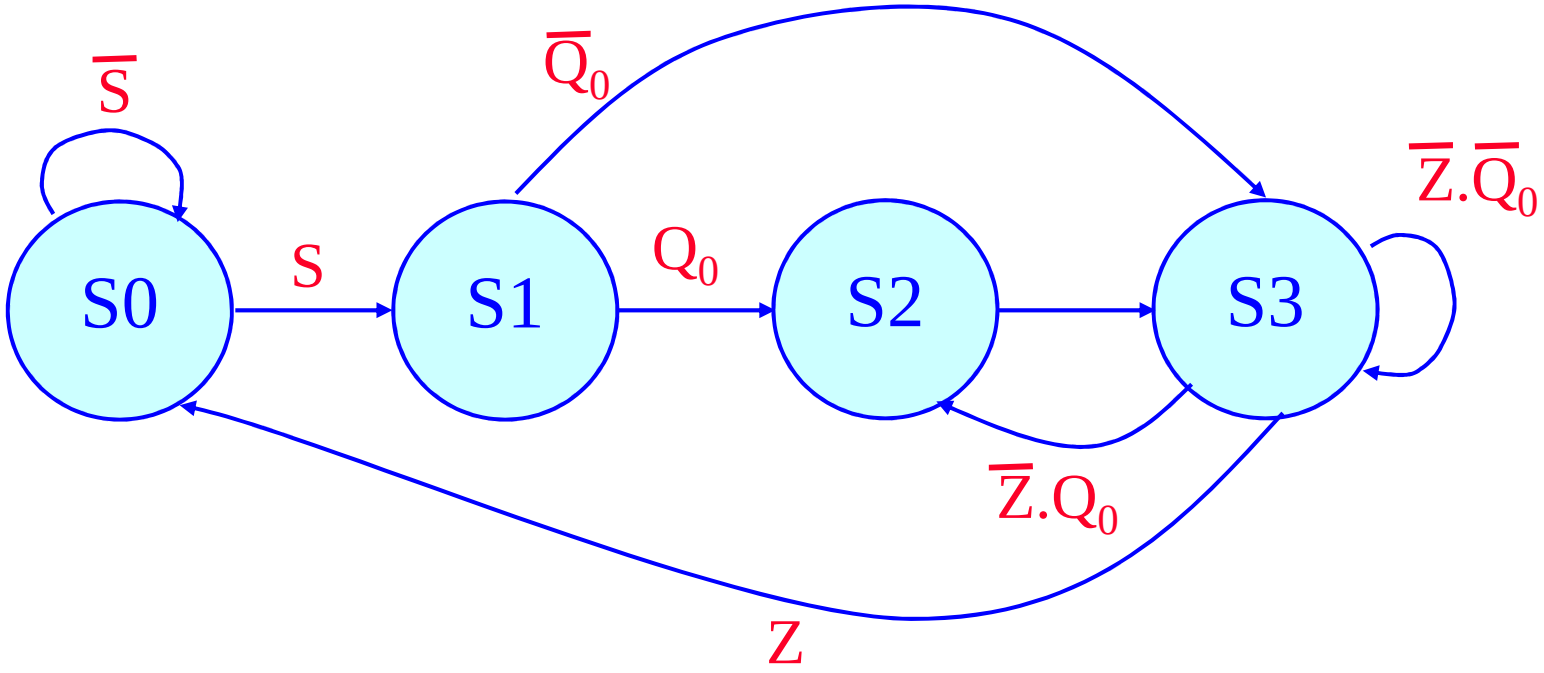
1. Hardwired Control: Binary Multiplier

- *Control unit: new ASM chart*



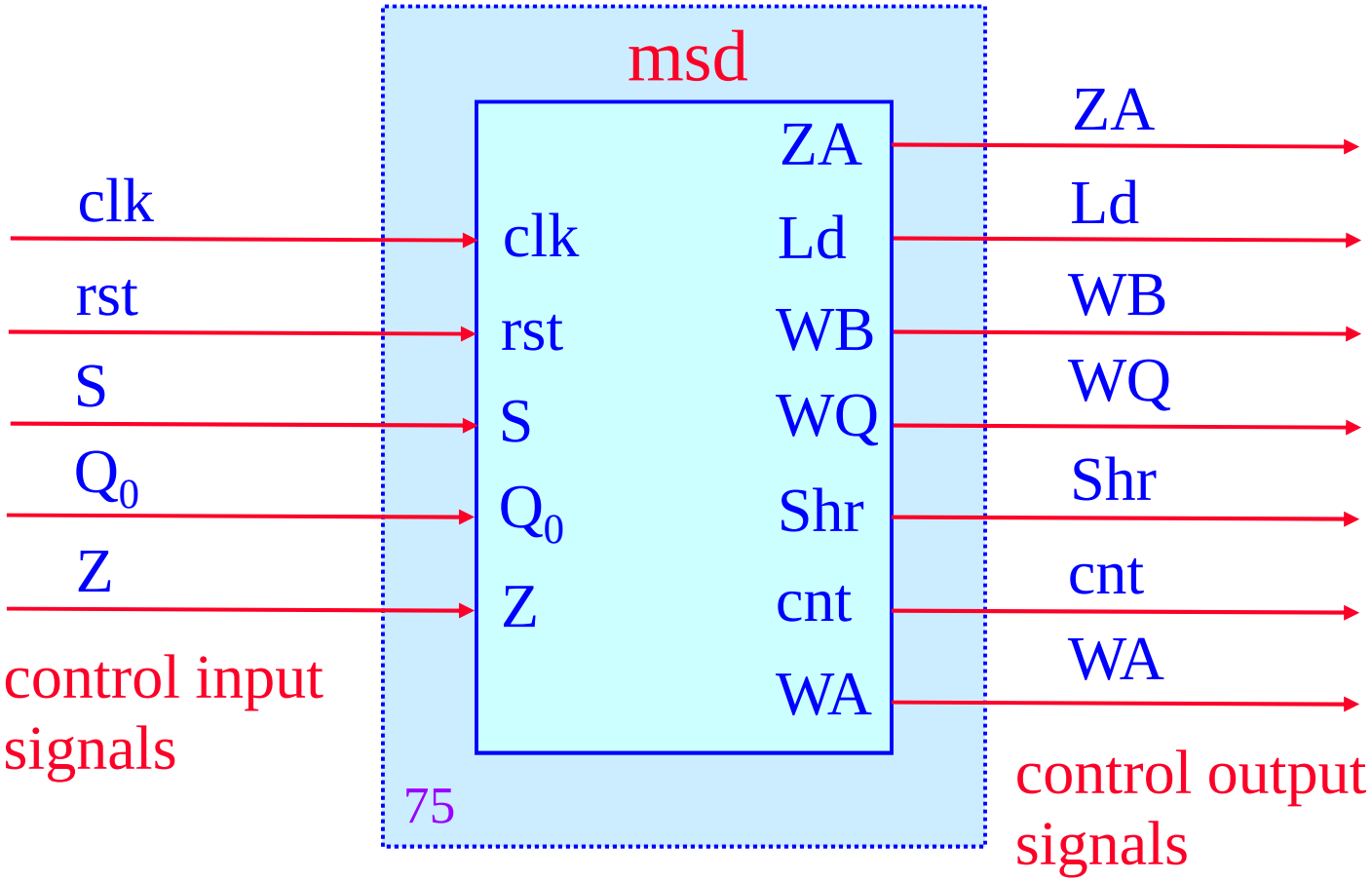
1. Hardwired Control: Binary Multiplier

- *Control unit: state diagram 2*



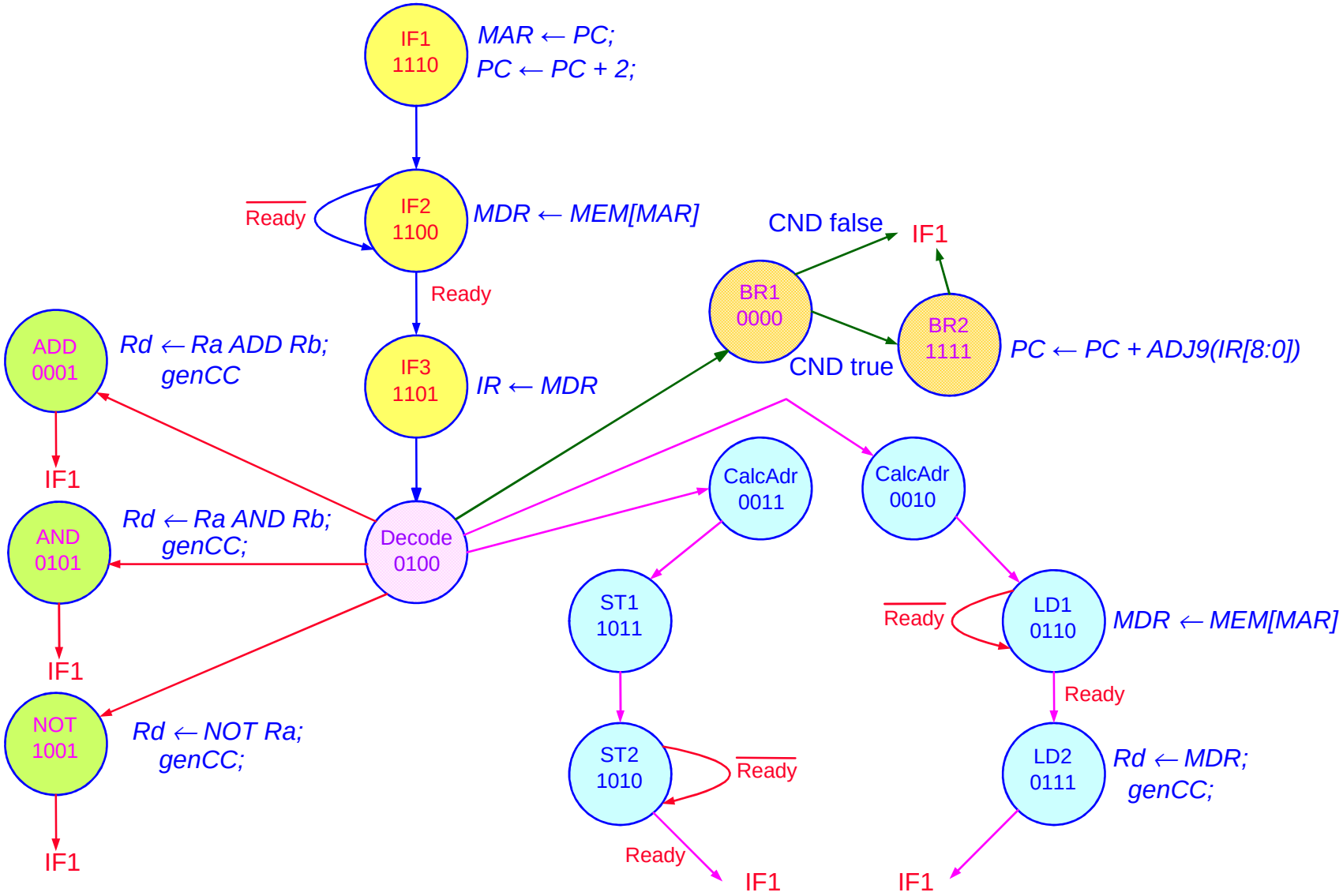
1. Hardwired Control: Binary Multiplier

- *Control unit: block diagram*

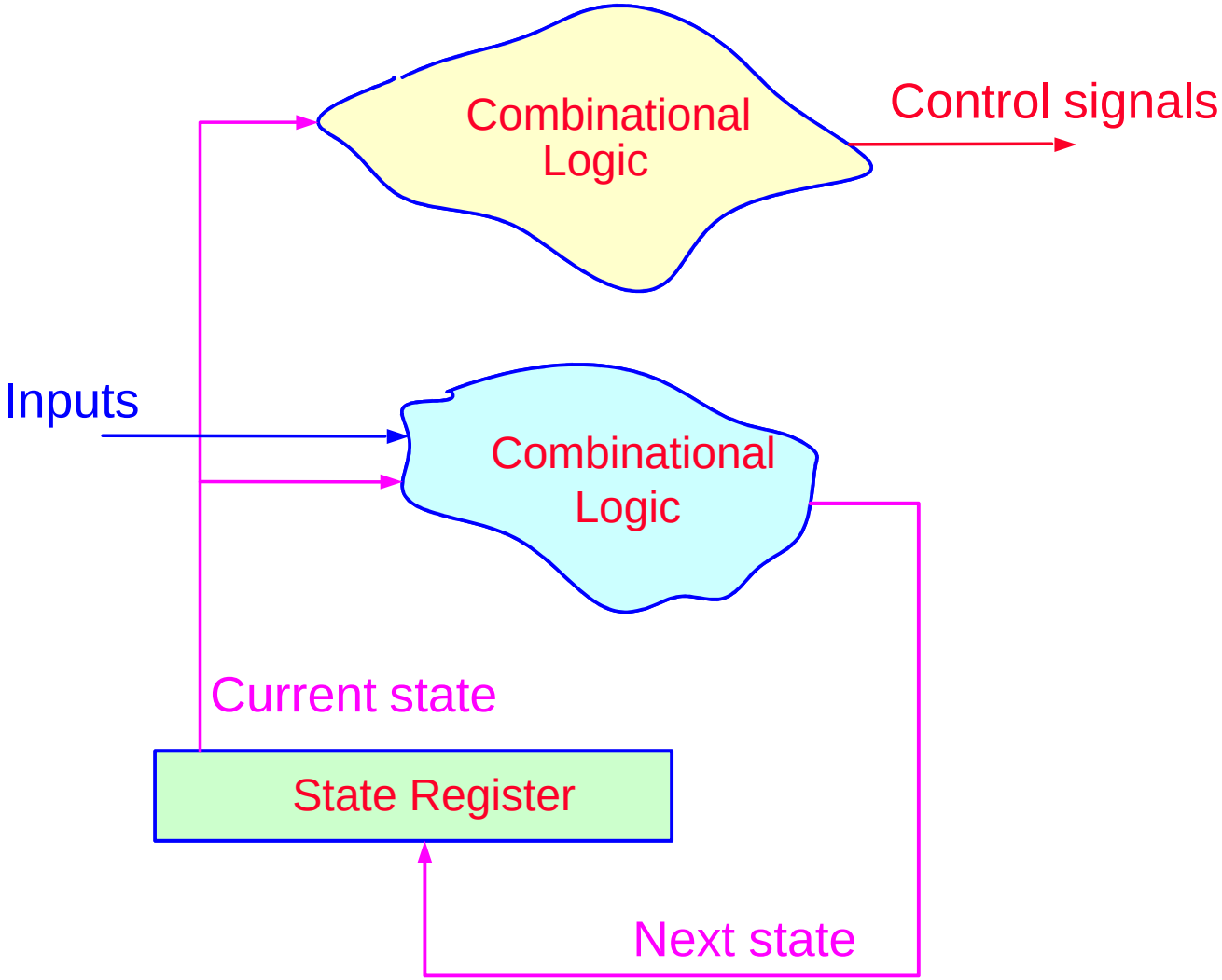


Microprogrammed Control

1. Control Logic State Diagram



2. Hardwired Control Units



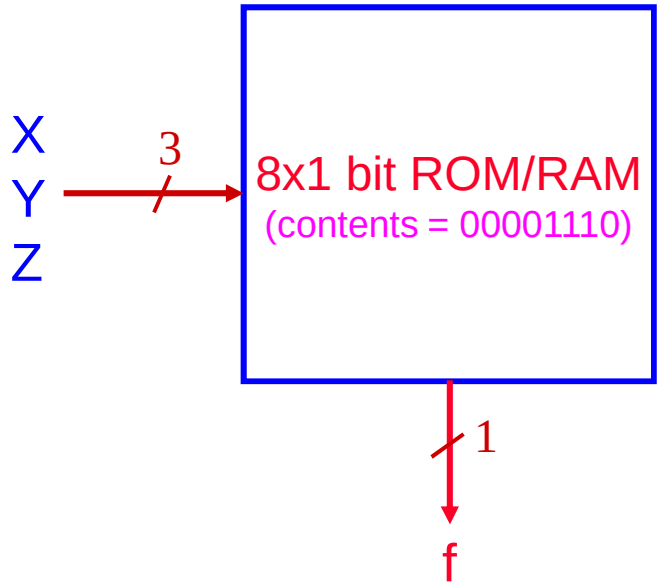
3. Microprogramming

- Observation: Much of the difficulty of designing or implementing hardwired control units comes from *the need to optimize* the next state generation logic
- Idea: If we could use a memory to hold the state transition diagram for the control unit, we wouldn't have to bother optimizing any logic
  - Use current state as "address" into memory
  - Contents of each memory location are the next state for each state
- Control units that use this technique are called microprogrammed control units

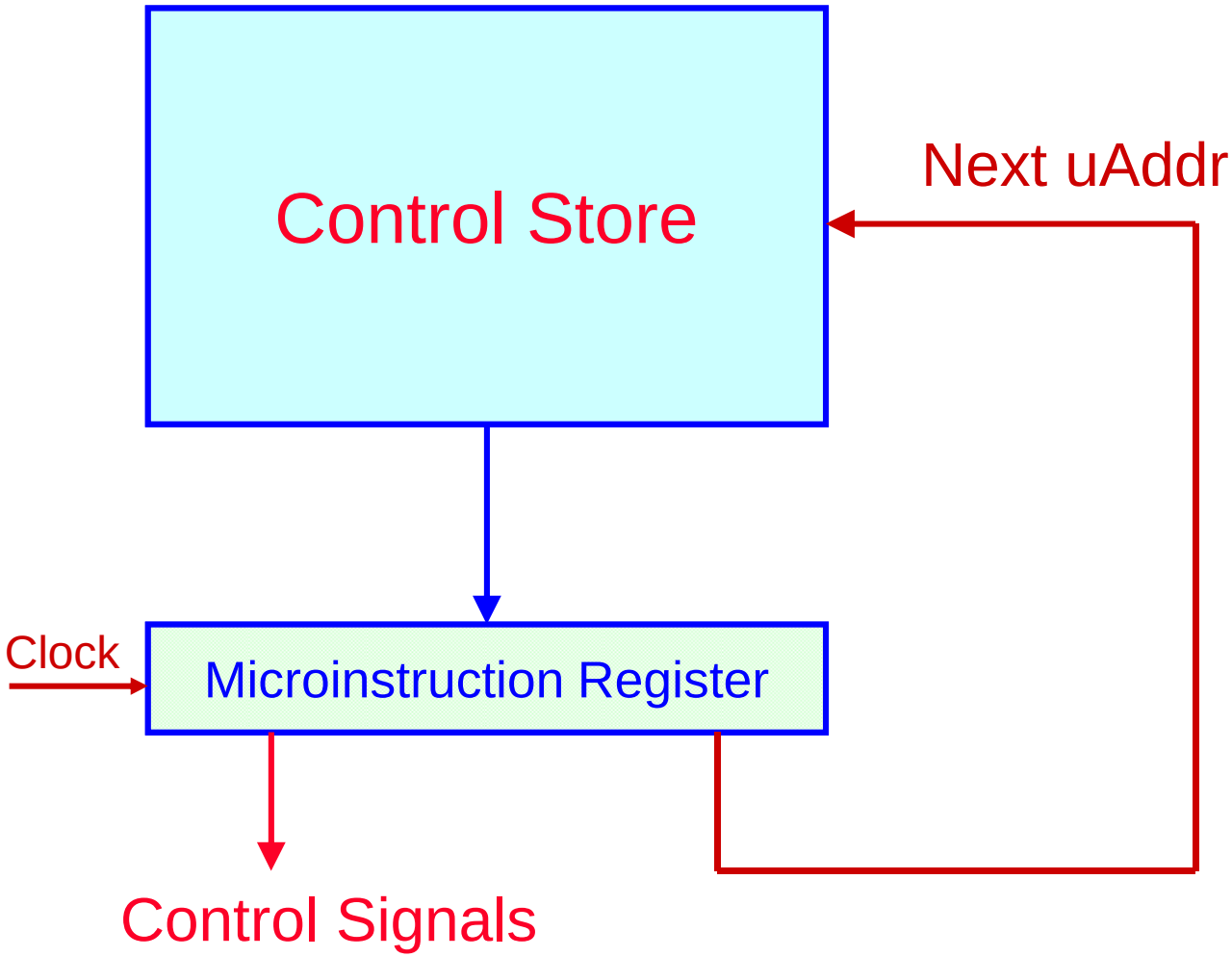
3.1 Using Memory to Implement Logic

$$f(x, y, z) = xy\bar{z} + x\bar{z}$$

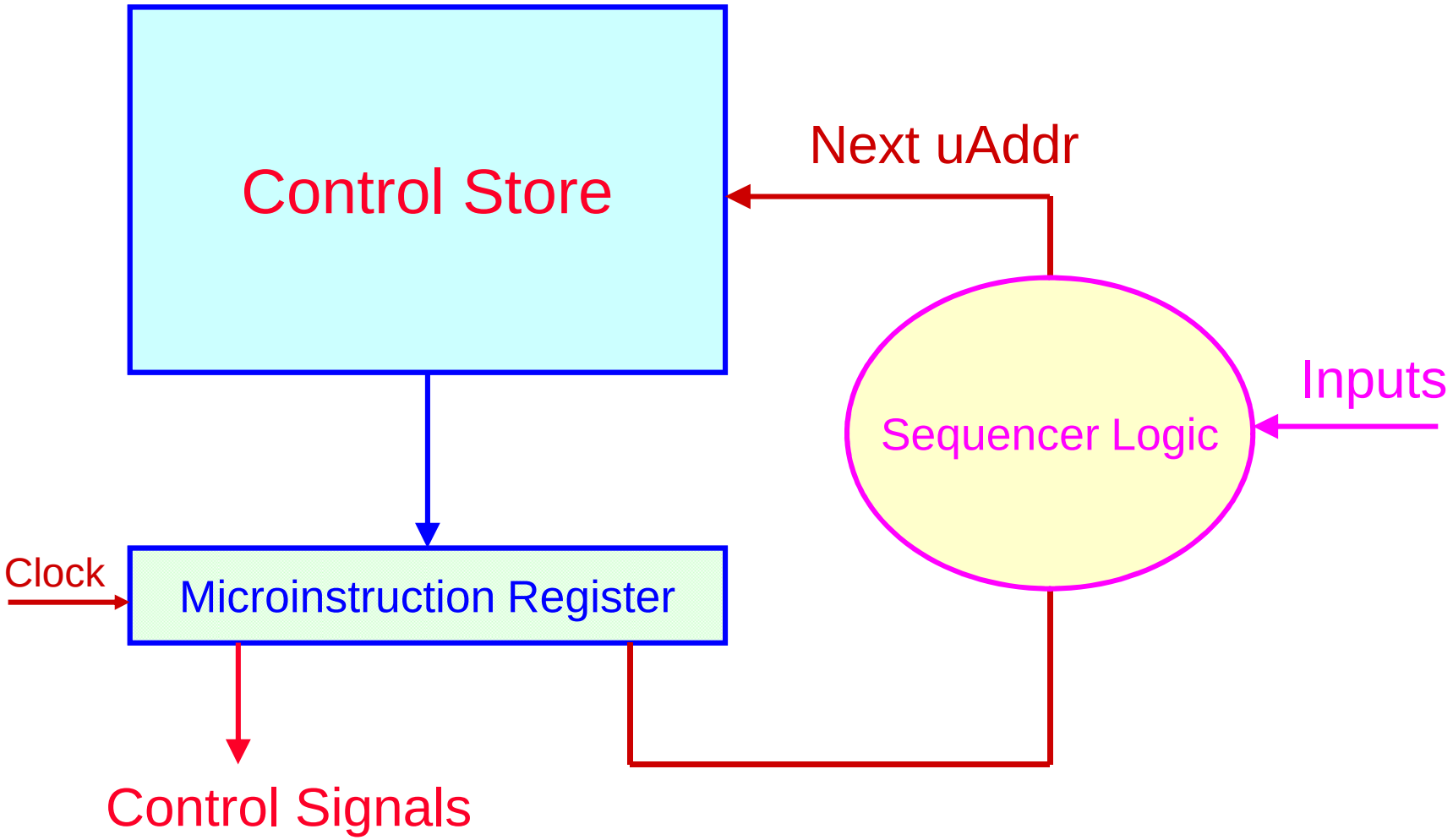
X	Y	Z	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0



3.2 Microprogrammed Control Unit



3.2 Microprogrammed Control Unit



3.2 Some Vocabulary

- Control store: the RAM or ROM that holds the truth table for a *microprogrammed control system*
- Microprogramming: the process of realizing a combinational circuit in a control unit based on ROM/RAM
- Microprogram/microcode: the contents of the control store in a *microprogrammed control system*
- Microinstruction: the contents of a single location in the *control store*