Introduction to digital systems and microprocessors

© Luis Entrena, Celia López, Mario García,
Enrique San Millán

Universidad Carlos III de Madrid

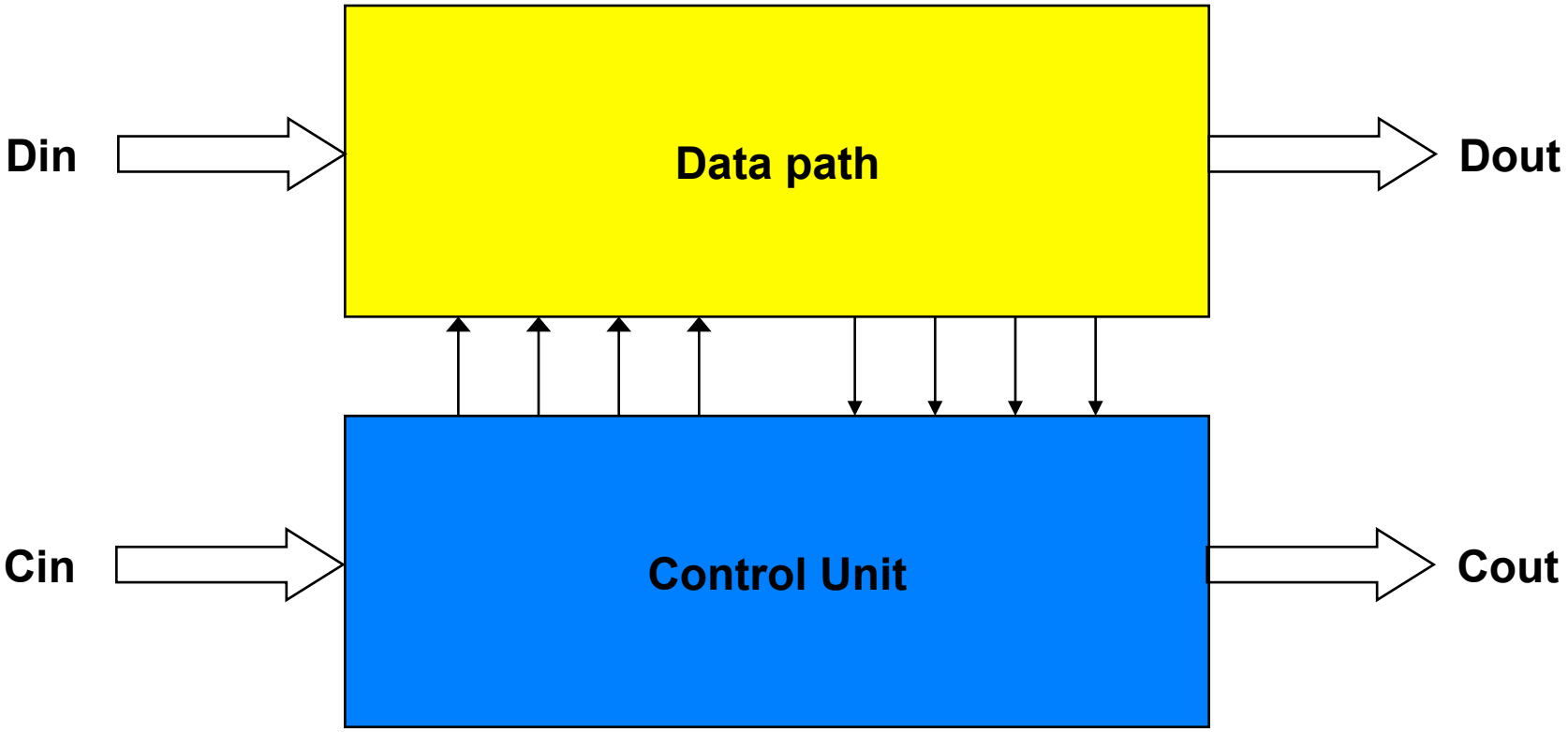
Outline

- Structure of a digital system
  - Data path
  - Control unit
- Structure of a basic computer
- Operation of a basic computer. Instructions

Digital systems

- A digital system processes digital information according to a particular algorithm
- Algorithm: a finite sequence of operations to solve a problem
- Goal of this lesson: introduce digital systems

Structure of a digital system



The Datapath

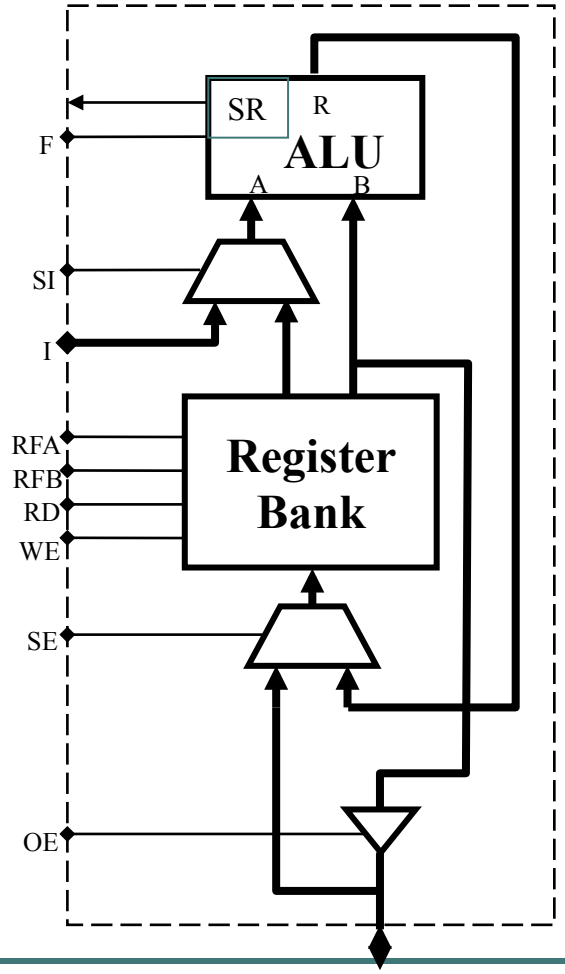
- Any algorithm can be decomposed in a set of basic operations:
  - Arithmetic
  - Logic
  - Shift & Rotation
- The datapath is the set of functional units that process the data

The Datapath

- Typical datapath components:
  - ALUs: perform operations
  - Registers and memories: store temporal data
  - Buses: connect datapath elements
  - Multiplexers: select data to be processed every time

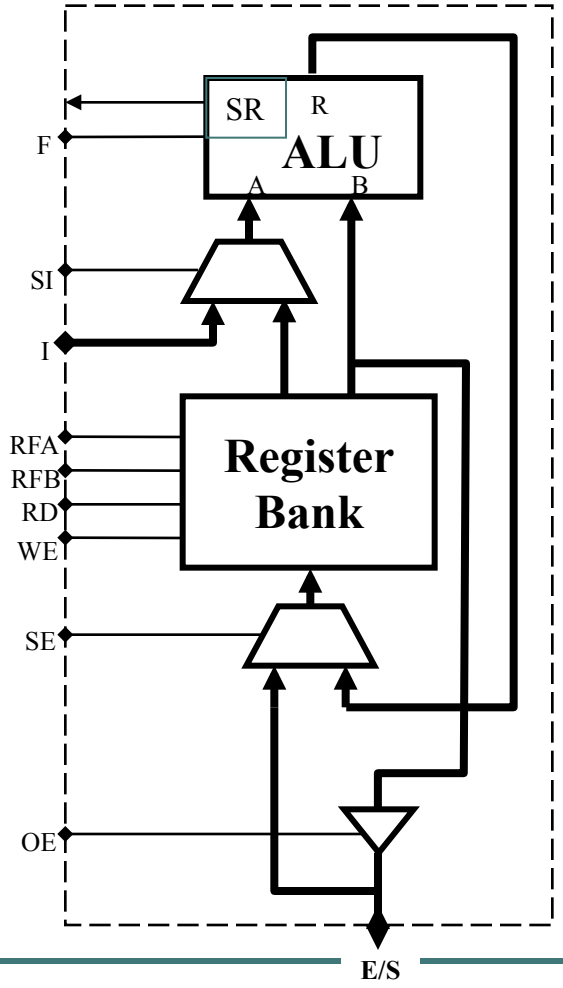
Datapath example

- ALU performs operations
  - F: ALU function
- SR (Status Register). Flag bits resulting from the ALU operation, such as :
  - C: Carry
  - O: Overflow
  - Z: Zero
  - S: Sign
- Immediate operands provided by I and selected by SI



Datapath example

- Register bank (triple port memory) stores temporal data
- Register selection
  - RFA: Source register A
  - RFB: Source register B
  - RD: Destination register
- I/O for external connection
  - SE selects RD input (external o internal)
  - OE (Output Enable)



The Control Unit

- Determines the correct application and sequencing of operations on data
- Typical Control Unit components:
  - Finite State Machines
  - Counters
  - Registers and memories to store control data, etc.

Control Unit example

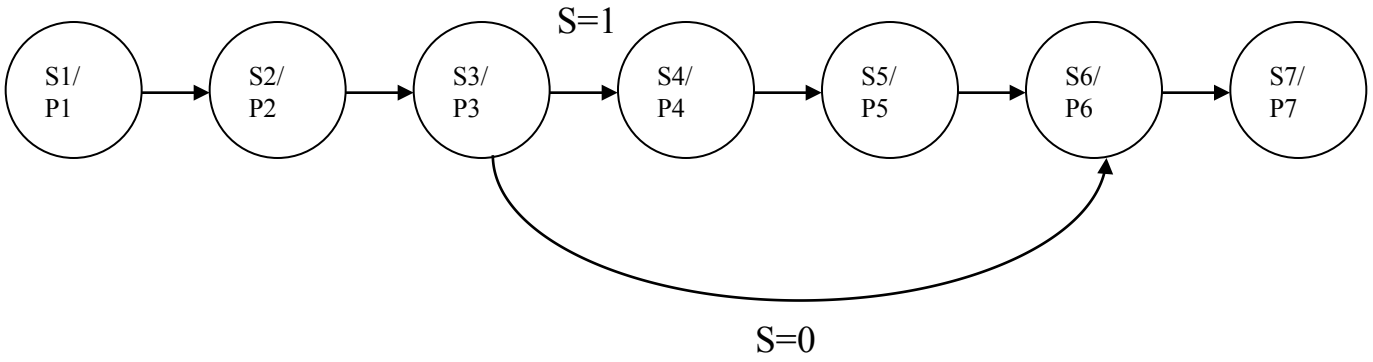
- Example: perform the following computation

  y = abs(x1 - x2)/2

- Steps:
  - 1. Load x1 in register R1
  - 2. Load x2 in register R2
  - 3. Subtract x2 from x1 and place the result in register R3 (R3 = R1 – R2).
    - If the result is positive, go to step 6. Otherwise, continue with step 4.
  - 4. Complement R3 (R3 = NOT R3)
  - 5. Increment R3 (R3 = R3 +1)
  - 6. Shift right R3 (R3 = R3/2)
  - 7. Send R3 to the output

Control Unit example

- FSM of the Control Unit



Control Unit example

- FSM outputs

Paso	A L U Function (F)	RFA	SI	I	RFB	RD	WE	SE	OE
1	X	X	X	X	X	R1	0	1	0
2	X	X	X	X	X	R2	0	1	0
3	RESTA	R1	0	X	R2	R3	0	0	0
4	NOT	R3	0	X	X	R3	0	0	0
5	SUMA	X	1	1	R3	R3	0	0	0
6	SHR	R3	0	X	X	R3	0	0	0
7	X	X	X	X	R3	X	1	X	1

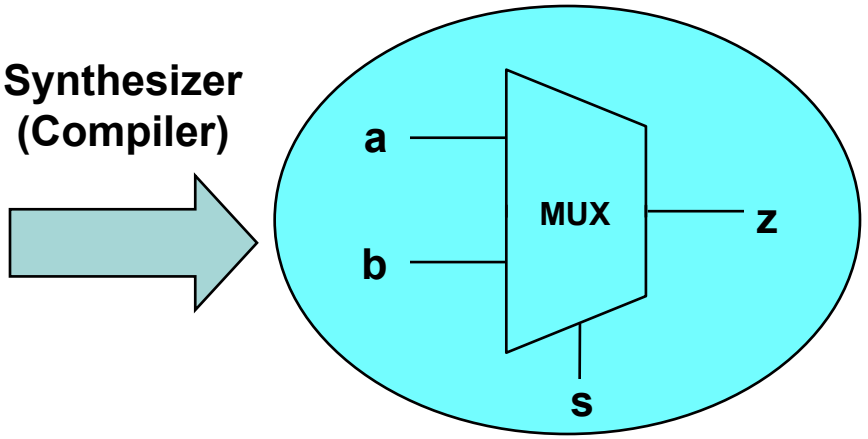
Digital systems design

- Designing with very basic components, such as logic gates, is not practical for digital systems design
- RTL (Register Transfer Level) design:
  - Use more abstract components, such as ALUs, registers or multiplexers
  - Digital systems are described by means of operations performed with data stored in registers

Hardware Description Languages

- Hardware Description Languages (HDLs) allow a designer to work at a higher abstraction level
- Example (VHDL)

```
if s = '0' then
        z <= a;
else
        z <= b;
end if;
```

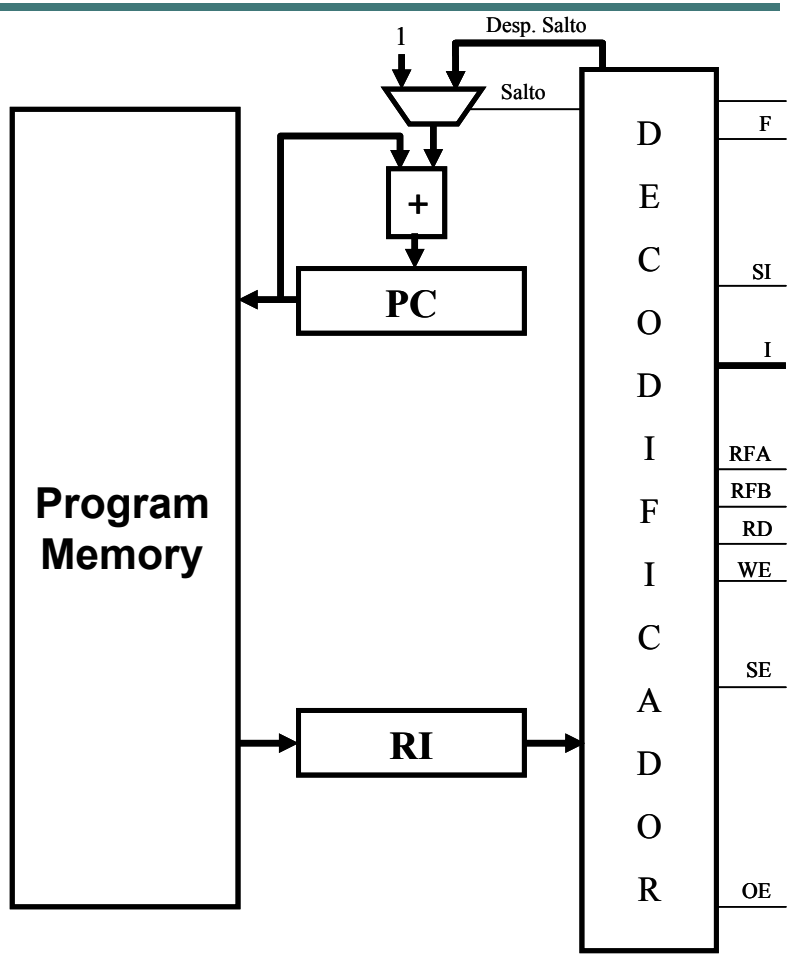


Structure of a basic computer

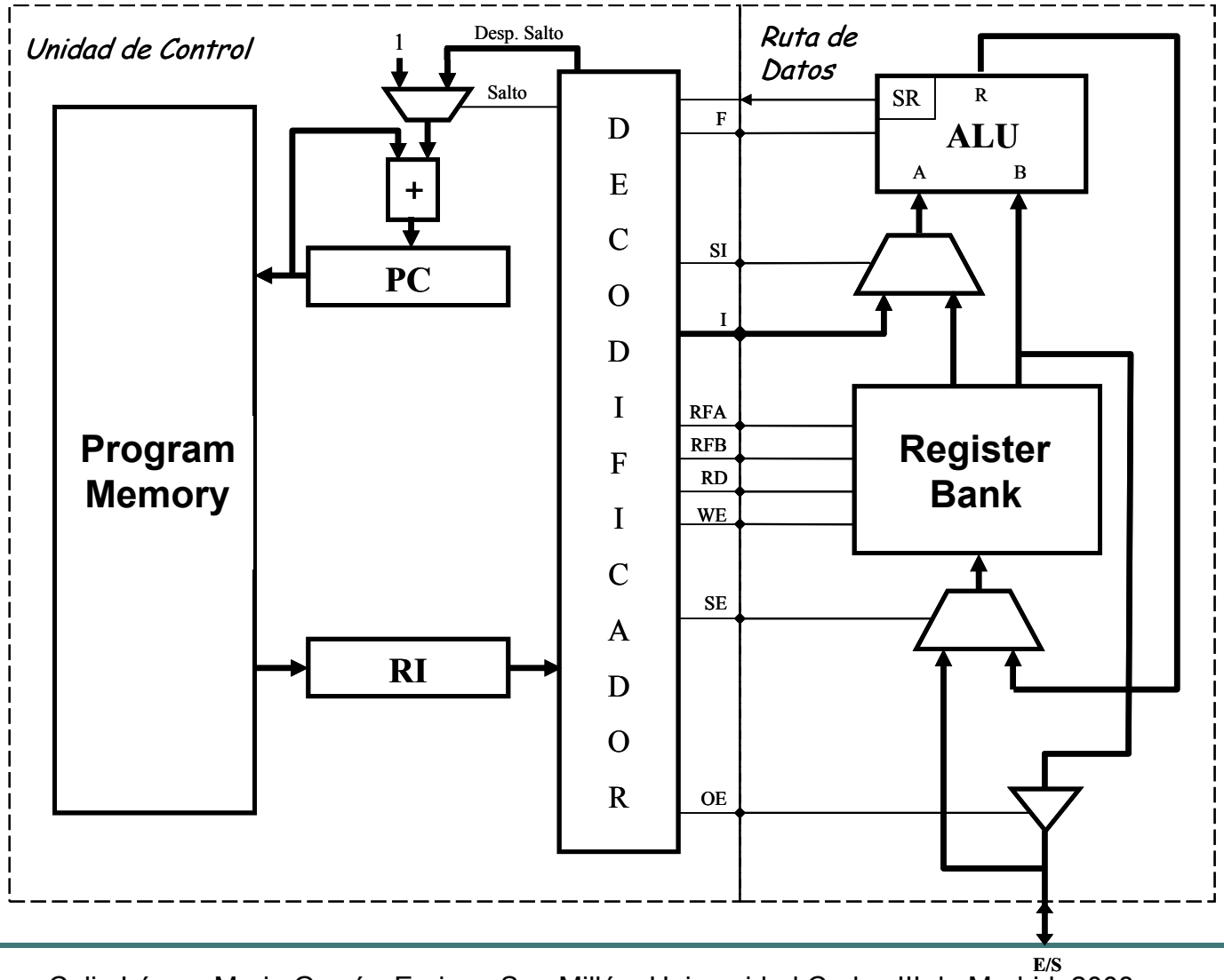
- Computer: general purpose machine for information processing
  - Oriented to implement a wide variety of algorithms
- Components:
  - Processing Unit or Central Processing Unit (CPU) performs the required operations
    - General purpose datapath
    - Programmable control unit, able to perform many differents algorithms
  - Memory to store information
  - Input/Output units for external communication

Programmable Control Unit

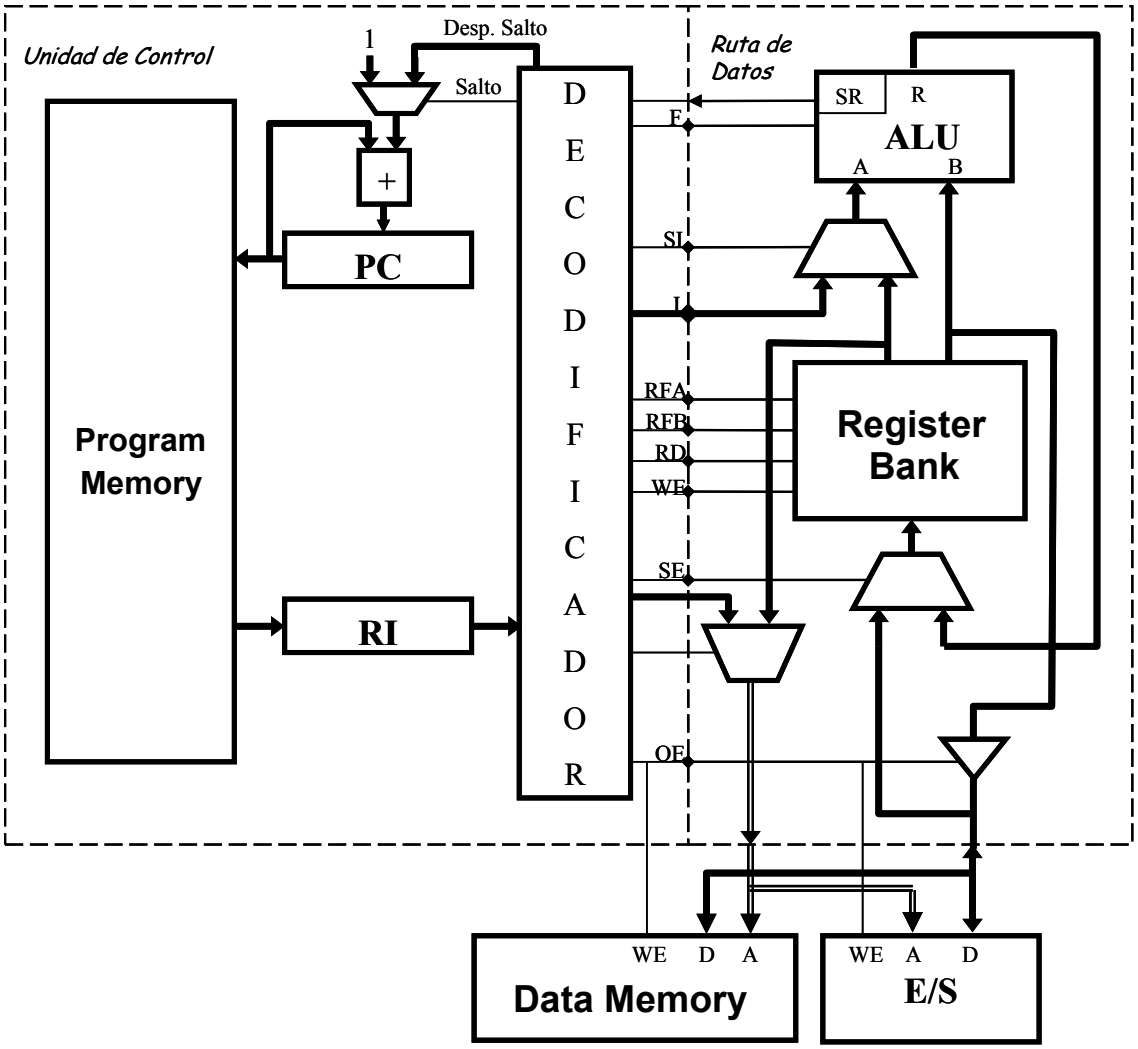
- Control words are stored in a memory
  - Control words are coded to save space
  - Coded control words are called *instructions*
- Every clock cycle, an instruction is read (RI) and decoded
- The instruction required for each step is determined by the Program Counter (PC).
  - Sequence changes are implemented by means of instructions, that can dynamically depend on the SR value



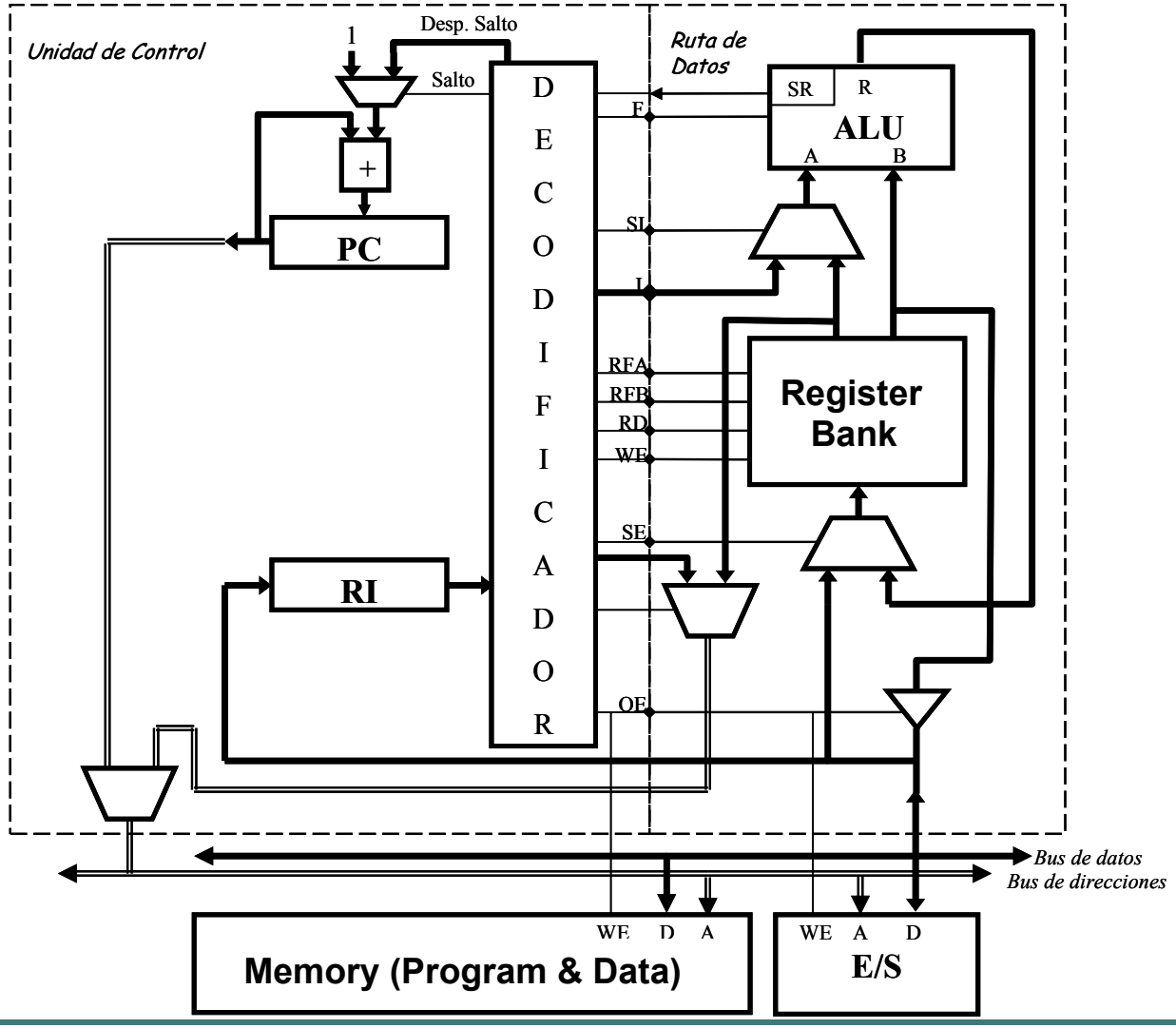
Microprocessor



Basic computer (Harvard architecture)



Basic computer
(von Neuman architecture)



Operation of the basic computer. Instructions

- How are instructions executed?
  - Instruction cycle
- What are the main types of instructions?
- How are instructions coded?
  - Instruction format
  - Addressing modes

Instruction cycle

- Every instruction is typically processed in two steps:
  - Instruction fetch: load instruction in IR
  - Instruction execution: decode instruction and configure the datapath to execute the operation specified by the instruction
- Each of these steps can be executed in a single clock cycle or in many clock cycles, depending on the microprocessor's complexity

Types of instructions

- Data transfer instructions
  - Transfer data between registers, register and memory, or register and I/O
- Arithmetic and logical instructions
  - Set ALU operations: add, subtract, shift, etc.
- Control flow instructions
  - Modify the instruction execution sequence
  - Modify PC contents
  - Examples: branches, subroutine calls, etc.

Instruction format

- Organization of instruction information. Instructions are typically field coded:
  - Operation code (Opcode): specifies the operation to be performed
  - Operands: specifies the data to be operated
- The number of operands and the size of each field may vary
- Instruction size must be an integer multiple of memory width

Addressing modes

- Operands can be specified in different ways, called addressing modes
- Some usual addressing modes:
  - *Immediate*: the operand is given with the instruction
  - *Register direct:* the instruction gives a register that contains the operand
  - *Memory direct:* the instruction gives a memory address that contains the operand
  - *Indirect*: the instruction gives a register that contains the memory address of the operand

Instruction format example

- Consider the previous microprocessor architecture, with the following parameters:
  - Opcode field: 5 bits
  - Register bank: 8 registers (direct addressing with 3 bits)
- Instruction format

Opcode	OpA	OpB	OpD	Other

Instruction examples

Operation	Mnemonic	Operation Code
Load data from memory	LD	0 0 0 0 0
Store data in memory	ST	0 0 0 0 1
Add	ADD	0 1 0 0 0
Subtract	SUB	0 1 0 0 1
NOT	NOT	0 1 1 0 0
AND	AND	0 1 1 0 1
OR	OR	0 1 1 1 0
Shift	SHL, SHR	0 1 1 1 1
Jump	JMP	1 0 0 0 0
Subroutine call	CALL	1 0 1 0 0
Subroutine return	RET	1 0 1 0 1
.	.	.

Instruction examples

- Single word instruction

Opcode	RFA	RFB	RD	Other
01000	001	010	011	00

R3 = R1 + R2

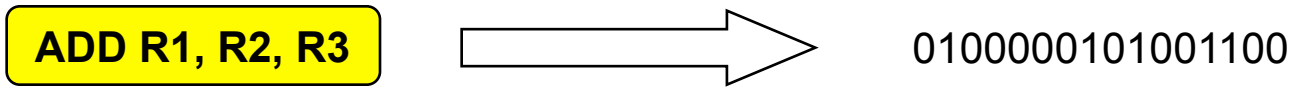
- Double word instruction

Opcode	RFA	RFB	RD	Other
00000	000	000	011	00
Direccion				
1010 1011 1100 1101				

Load R3 with data in memory position ABCDh

Assembler language

- Binary instruction codes are cumbersome
- *Assembly Language:* instructions are specified as mnemonics and operands are specified using symbolic names
- Assembly Program: translate instructions in a program to their equivalent binary codes
- Example:



References

- “Principios de Diseño Digital”. D. Gajski. Ed. Prentice Hall