

**UNIT 2.FUNCTIONAL ELEMENTS OF A COMPUTER**

Activity 2. Solution

**Computer Systems**

**CFGS DAW**

Autores: Alfredo Oltra / Sergio Garcia

Adaptado: Vicent Bosch [vicent.bosch@ceedcv.es](mailto:vicent.bosch@ceedcv.es)

2020/2021

Versión:201023.1329

Licencia

**Reconocimiento – NoComercial – CompartirIgual (by-nc-sa)**: No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

## Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

 Importante

 Atención

 Interesante

UD01. FUNCTIONAL ELEMENTS OF A COMPUTER

Activities 2. Solutions

1. **EXERCISE 1**

**(Exercise 1)** We have a hypothetical computer with this instruction format:

|  |  |  |
| --- | --- | --- |
| OP\_CODE | OPERAND 1 | OPERAND 2 |

4 BITS 4 BITS 4 BITS



0000

0xC2

0001

0x19

0010

0x5A

0011

0x2

...

*Figure 1. Memory (address and content)*

SUM Rx, Ry 1001xxyy Add RX+RY and it is stored in RX.

Following the instruction sequence:

**100100010010**

1. What is the result after executing this instruction?

**The UC decodes the instruction according to the instruction format:100100010010**

|  |  |  |
| --- | --- | --- |
| **1001** | **0001** | **0010** |

4 BITS 4 BITS 4 BITS

**La instrucción es agregar lo que está almacenado en 0001 (dirección 1) y en 0010 (dirección 2)**

**Si comprobamos la unidad de memoria, en la dirección 0001 el valor almacenado es 0x19 y en la dirección 0010 el valor almacenado es 0x5A. Dado que los valores se representan usando hexadecimal, debemos convertir a binario y realizar la operación de suma. En la CPU, esta operación la realiza la ALU.**

19-> 0 0 0**1** 1**1** 1 0 0 1

5A->0 1 0 1 1 0 1 0

0 1 1 1 **0** 0 1 1 -> **73 16**

1. Which will be the state of the memory after the execution of this instruction?

**Dado que la instrucción add dice Add RX + RY y se almacena en RX. Esto significa que el resultado de la operación de adición debe almacenarse en la dirección RX, que en este caso es la dirección 0001. Allí se sobrescribe el valor.**



...

|  |  |
| --- | --- |
| 0000 | 0xC2 |
| 0001 | ***0x73*** |
| 0010 | 0x5A |
| 0011 | 0x2 |

1. Which is the addressing mode used in both operands?

**Ambos operandos almacenan la dirección del contenido que será utilizado por la instrucción add, por lo tanto, el modo de direccionamiento es absoluto / directo (página 11 Unidad 2)**

1. What would be the result if operand 2 uses immediate addressing mode?

**Cuando se usa el modo de direccionamiento inmediato, el contenido se almacena en el propio campo del operando, por lo que la operación de adición sería:**

19→ 0 0 0 1 1 0 0 1

2→ 0 0 0 0 0 0 1 0

0 0 0 1 1 0 1 1→ **1B 16**

# EXERCISE 2

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Memory | | | | | | | | | | | | | | | | | | |
| 0 |  |  |  |  |  |  |  |  |  | 16 |  |  |  |  |  |  |  |  |
| 1 |  |  |  |  |  |  |  |  | 17 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 2 |  |  |  |  |  |  |  |  | 18 |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |  | 19 |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |  | 20 |  |  |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |  | 21 |  |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |  | 22 |  |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |  | 23 |  |  |  |  |  |  |  |  |
| 8 |  |  |  |  |  |  |  |  | 24 |  |  |  |  |  |  |  |  |
| 9 |  |  |  |  |  |  |  |  | 25 |  |  |  |  |  |  |  |  |
| 10 |  |  |  |  |  |  |  |  | 26 |  |  |  |  |  |  |  |  |
| 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 27 |  |  |  |  |  |  |  |  |
| 12 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 28 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 13 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 29 |  |  |  |  |  |  |  |  |
| 14 |  |  |  |  |  |  |  |  | 30 |  |  |  |  |  |  |  |  |
| 15 |  |  |  |  |  |  |  |  | 31 |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Registers | | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 3 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

**00001011** Write in memory position 11

(A) [01010101] Write 1

**00001100** Write in memory position 12

(B) [00100001] Write 2

**00010001** Write in memory position 17

(C) [00000010] Write 3

**00011100** Write in memory position 28

(D) [00101101] Write 4

**01001011** Copy the data from memory position 11 to register 0 **10000100** Copy the data from Register 0 to Register\_1: 1 → in R1 **01011100** Copy the data from memory position 28 to register 0 **10001100** Copy the data from Register 0 to Register\_3: 4 → in R3 **01010001** Copy the data from memory position 17 to register 0 **10001000** Copy the data from Register 0 to Register\_2: 3 → in R2

**10111110** Multiply the content of R3 and R2 and write the result in R3 [3\*4] → 12 in R3

**10101101** Subtract the content of R3 and R1 and write the result in R3 [12‐1] → 11 in R3

**01001100** Copy the data from memory position 12 to Register\_0

**10001000** Copy the data from Register 0 to Register\_2: 2 → in R2

**10011110** Add the content of R3 and R2 and write the result in R3 [2 + 11] → 13 ¡n R3

**01010001** Copy the data from memory position 17 to Register\_0 **10001000** Copy the data from register\_0 to register\_2 → 3 in R2 **11001110** Divide the content of R3 by R2 and write in R3 [13/3] 4 in R3 **10000011** Copy the date from R3 to R0 → 4 in R0

**01101101** Write in memory position 13 the content of Register 0

**00101101** Show in the screen the content of memory position 13

* 1. ***Solution***

1. Formula: ((D\*C)‐A+B)/C
2. 4 (Content of memory position 13)
3. The state shown in the solution
4. If the PC was initially at 258 and we have executed 21 instructions, the PC will contain the value 279
5. we have two bits, i. e. 4 registers.