

# UNIT 2. FUNCTIONAL ELEMENTS OF A COMPUTER Activities-4 (review)

Computer Systems CFGS DAW

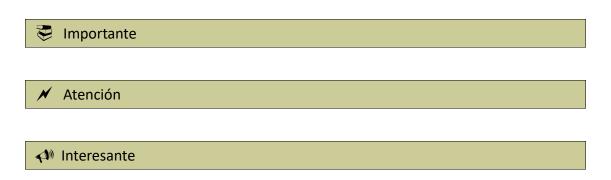
Vicent Bosch vicent.bosch@ceedcv..es 2020/2021 Versión:210109.1000

#### Licencia

Reconocimiento - NoComercial - Compartirlgual (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:



# UD02. FUNCTIONAL ELEMENTS OF A COMPUTER Activities-4

**(Exercise 1)** We have a hypothetical computer with this instruction set. Each character in the instruction field corresponds to a bit.

Code	Instrucction	Description
LOAD RX, MMMM	00rxmmmm	Loads content of memory
		mmmm in Register rx
STORE MMMM, RX	01rxmmmm	Stores content of
		Register rx in memory mmmm
ADDi RX,RY	1000rxry	Performs $rx+ry$ and sends the
		result to the register <b>R1</b>
SUBi RX,RY	1100rxry	Performs $rx-ry$ and sends the
		result to the register R2
MULTi RX,RY	1111rxry	Performs $rx*ry$ and sends the
		result to the register ${m r}{m x}$

The memory has the following information (numbers are in binary representation):

<i>Address</i>	Content
0000	
0001	
0010	
0011	
0100	
0101	
0110	
0111	
1000	
1001	
1010	00000111
1011	00001111
1100	
1101	
1110	00100001
1111	00000110

Content
00000000
00000000
00000000

And the following instructions of a program to be executed (numbers are in hexadecimal representation):

i1: LOAD R1, #A
i2: LOAD R2, #F
i3: ADDi R1, R2
i4: STORE #5, R1
i5: MULTi R1, R3
i6: MULTi R2, R3
i7: LOAD R1, #B
i8: LOAD R2, #E
i9: SUBi R2, R1
i10: STORE #4, R2

Execute each instruction and update the values of registers and memory addresses and their content. It is recommended to resolve these types of exercises using pen and paper, and without a calculator 

::

Remember, MAR and MDR registers are used to access the memory and get/save data.

#### i1: LOAD R1, #A

Get the data from address #A (1010 in binary) and sends it to register R1

Address	Content	Register	Content
0000		R1	00000111 <
0001		R2	
0010		R3	
0011			
0100			
0101			
0110			
0111			
1000			
1001			
1010	00000111		
1011	00001111		
1100			
1101			
1110	00100001		
1111	00000110		

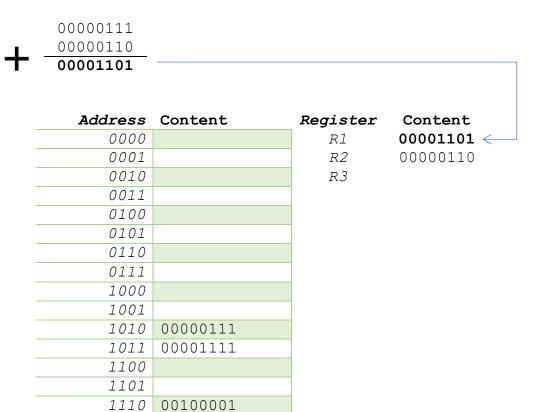
#### i2: LOAD R2, #F

Get the data from address #A (1010 in binary) and sends it to register R2

<i>Address</i>	Content	Register	Content
0000		R1	00000111
0001		R2	00000110 <
0010		R3	
0011			
0100			
0101			
0110			
0111			
1000			
1001			
1010	00000111		
1011	00001111		
1100			
1101			
1110	00100001		
1111	00000110 —		

#### i3: ADDi R1,R2

The ALU performs the addition R1+R2 and sends the result to R1



1111 00000110

#### i4: STORE #5, R1

<i>Address</i>	Content	Register	Content
0000		R1	00001101
0001		R2	00000110
0010		R3	
0011			
0100			
0101	00001101		
0110			
0111			
1000			
1001			
1010	00000111		
1011	00001111		
1100			
1101			
1110	00100001	1	
1111	00000110	1	

#### i5: MULTi R1, R3

R1xR3 and sends the result to R1 (check how to multiply in Page 9. Unit 1)

00001101 00000000 **00000000** 

Address	Content
0000	
0001	
0010	
0011	
0100	
0101	00001101
0110	
0111	
1000	
1001	
1010	00000111
1011	00001111
1100	
1101	
1110	00100001
1111	00000110

Register	Content
R1	0000000
R2	00000110
R3	0000000

#### i6: MULTi R2, R3

R2xR3 and sends the result to R3 (check how to multiply in Page 9. Unit 1)

00000110 00000000 **00000000** 

Address	Content
0000	
0001	
0010	
0011	
0100	
0101	00001101
0110	
0111	
1000	
1001	
1010	00000111
1011	00001111
1100	
1101	
1110	00100001
1111	00000110

# Register Content R1 00000000 R2 00000000 ← R3 00000000

#### 17: LOAD R1, #B

Get the data from address #B (1011 in binary) and sends it to register R1

Address	Content	Register	Content
0000		R1	00001111 <
0001		R2	
0010		R3	
0011			
0100			
0101	00001101		
0110			
0111			
1000			
1001			
1010	00000111		
1011	00001111 —		
1100			
1101			
1110	00100001		
1111	00000110		

#### i8: LOAD R2, #E

### Get the data from address #E (1110 in binary) and sends it to register R2

Address	Content	Register	Content
0000		R1	00001111
0001		R2	00100001
0010		R3	
0011			
0100			
0101			
0110			
0111			
1000			
1001			
1010	00000111		
1011	00001111		
1100			
1101			
1110	00100001		
1111	00000110 —		

#### I9: SUBi R2, R1

#### R2-R1 and sends the result to register R2

(check how to subtract in Unit 1)

00100001 00001111 00010010

<i>Address</i>	Content
0000	
0001	
0010	
0011	
0100	
0101	
0110	
0111	
1000	
1001	
1010	00000111
1011	00001111
1100	
1101	
1110	00100001
1111	00000110

Register	Content		
R1	00001111		
R2	00010010		
R3			

#### i10: STORE #4, R2

## Sends the content of register R2 to the memory address #4 (0100)

<i>Address</i>	Content	Register	Content
0000		R1	00001111
0001		R2	00010010 ———
0010		R3	
0011			
0100	00010010		
0101			
0110			
0111			
1000			
1001			
1010	00000111		
1011	00001111		
1100			
1101			
1110	00100001		
1111	00000110		