CENTRE ESPECÍFIC
D´EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

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

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 ***R1***
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 ***rx***

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

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

Register	**Content**
R1	00000000
R2	00000000
R3	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
0000	
0001	
0010	
0011	
0100	
0101	
0110	
0111	
1000	
1001	
1010	00000111
1011	00001111
1100	
1101	
1110	00100001
1111	00000110

Register	Content
R1	00000111
R2	
R3	

i2: LOAD R2, #F

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

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

Register	Content
R1	00000111
R2	00000110
R3	

i3: ADDi R1,R2

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

```
  00000111
+ 00000110
----------
  00001101
```

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

Register	Content
R1	**00001101**
R2	00000110
R3	

i4: STORE #5, R1

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	00001101
R2	00000110
R3	

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	**00000000**
R2	00000110
R3	00000000

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

I7: LOAD R1, #B

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

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	00001111
R2	
R3	

i8: LOAD R2, #E

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

Address	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	00100001
R3	

I9: SUBi R2, R1

R2-R1 and sends the result to register R2
(check how to subtract in Unit 1)

```
00100001
00001111
--------
00010010
```

Address	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)

Address	Content
0000	
0001	
0010	
0011	
0100	**00010010**
0101	
0110	
0111	
1000	
1001	
1010	00000111
1011	00001111
1100	
1101	
1110	00100001
1111	00000110

Register	Content
R1	00001111
R2	00010010
R3