UNIT 2.
FUNCTIONAL ELEMENTS OF A COMPUTER
Activities-3 *Solutions*

Computer Systems
CFGS DAW

Vicent Bosch
vicent.bosch@ceedcv..es
2020/2021
Versión:201103.1140

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-3

(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*
ADD RX,RY	1000rxry	Performs *rx+ry* and sends the result to the register ***R3***
SUB RX,RY	1100rxry	Performs *rx-ry* and sends the result to the register ***R3***
JNZ MMMM	1111mmmm	If the content of ***R3*** is not zero the ***PC*** jumps to memory *mmmm*

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

Address	**Content**
0000	
0001	
0010	
0011	00001100
0100	00001001
0101	00001101
0110	00001000
0111	
1000	
1001	
1010	
1011	
1100	
1101	
1110	
1111	

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

```
i1: LOAD R0, 3
i2: LOAD R1, 4
i3: ADD R0, R1
i4: STORE 10, R3
i5: LOAD R0, 5
i6: LOAD R1, 6
i7: SUB R0, R1
i8: JNZ 14 (i10) // memory address 14 stores instruction 10
i9: STORE 7, R3
i10: STORE 8, R3
```

a) Execute each instruction and update the values of registers and memory addresses and its content. For example, for the first instruction:

i1: R0→ 00001100 (8_{10})
No memory addresses/content updated

Instruction 2: `LOAD R1, 4`

R1 is a register, and number 4 is an address. Therefore, the UC (using MAR) accesses through the memory bus to the address 4 and send its content (`00001001`) to the MDR and after that the content is stored in register R1.

R1 now contains 00001001

Register	Content
R0	00001100
R1	**00001001**
R2	
R3	

Instruction 3: `ADD R0, R1`

The UC sends the contents of R0 and R1 to the ALU. The ALU performs an addition and sends the result to register R3.

R3 now contains 00010101

Register	Content
R0	00001100
R1	00001001
R2	
R3	**00010101**

Instruction 4: `STORE 10, R3`

The number 10 is a memory where the content of R3 will be stored.

Address	Content
…	
0011	00001100
0100	00001001
0101	00001101
0110	00001000
0111	
1000	
1001	
1010	**00010101**
1011	
…	

Instruction 5: `LOAD R0, 5`

R0 is a register, and number 5 is an address. Therefore, the UC (using MAR) accesses through the memory bus to the address 5 and send its content (`00001101`) to the MDR and after that the content is stored in register R0.

R0 now contains 00001101. Its previous content is overwritten.

Register	Content
R0	**00001101**
R1	00001001
R2	
R3	00010101

Instruction 6: `LOAD R1, 6`

R1 is a register, and number 6 is an address. Therefore, the UC (using MAR) accesses through the memory bus to the address 6 and send its content (`00001000`) to the MDR and after that the content is stored in register R1.

R1 now contains 00001000. Its previous content is overwritten.

Register	Content
R0	00001101
R1	**00001000**
R2	
R3	00010101

Instruction 7: `SUB R0, R1`

The UC sends the contents of R0 and R1 to the ALU. The ALU performs a subtraction (R0-R1) and sends the result to register R3.

R3 now contains 00000011

Register	Content
R0	00001101
R1	00001000
R2	
R3	**00000101**

Instruction 8: `JNZ 14`

This instruction can alter the sequentiality of the program since it can change the value of the PC. It checks the value of R3 and if the value is NOT zero then sends the content of memory address 14 to the PC. Since **R3 contains 00000011** the PC is changed and next instruction to execute is **instruction 10.**

Register	Content
PC	instruction 10

Instruction 9: `Not executed.`

Instruction 10: `STORE 8, R3`

The number 8 is a memory where the content of R3 will be stored.

Address	Content
…	
0011	00001100
0100	00001001
0101	00001101
0110	00001000
0111	
1000	**00000101**
1001	
1010	00010101
1011	
…	

b) Convert each instruction to binary representation.

```
i1: LOAD R0, 3      (00rxmmmm) → 00000011
i2: LOAD R1, 4      (00rxmmmm) → 00010100
i3: ADD R0, R1      (1000rxry) → 10000001
i4: STORE 10, R3    (01rxmmmm) → 01111010
i5: LOAD R0, 5      (00rxmmmm) → 00000101
i6: LOAD R1, 6      (00rxmmmm) → 00010110
i7: SUB R0, R1      (1100rxry) → 11000001
i8: JNZ 14 (i10)    (1111mmmm) → 11111110
i9: STORE 7, R3     (01rxmmmm) → 01110111
i10: STORE 8, R3    (01rxmmmm) → 01111000
```

c) Modify the content of address 0110 to store 00001101. Has this update any effect on the execution of the program?

Address	Content
0000	
0001	
0010	
0011	00001100
0100	00001001
0101	00001101
0110	**00001101**
0111	

This update alters the execution of instruction 6 which uses the memory address 6.

Instruction 6: `LOAD R1, 6`

R1 is a register, and number 6 is an address. Therefore, the UC (using MAR) accesses through the memory bus to the address 6 and send its content (**00001101**） to the MDR and after that the content is stored in register R1.

R1 now contains 00001101. Its previous content is overwritten.

Register	**Content**
R0	00001101
R1	**00001101**
R2	
R3	00010101

When the next instruction (**i7**) is executed (sub), the result of the ALU operation will be a zero.

Instruction 7: `SUB R0, R1`

The UC sends the contents of R0 and R1 to the ALU. The ALU performs a subtraction (R0-R1) and sends the result to register R3.

R3 now contains 00000011

Register	**Content**
R0	00001100
R1	00001001
R2	
R3	**00000000**

Next instruction (**i8**) will **NOT** change the value of PC since the content of R3 is zero.

Instruction 8: `JNZ 14`

This instruction can alter the sequentiality of the program since it can change the value of the PC. It checks the value of R3 and if the value is NOT zero then sends the content of memory address 14 to the PC. Since **R3 contains 00000000** the PC is not changed and next instruction to execute is **instruction 9.**

Instructions 9 and 10 are executed and the memory content will be this:

Address	**Content**
…	
0011	00001100
0100	00001001
0101	00001101
0110	00001000
0111	**00000000**
1000	**00000000**
1001	
1010	00010101

d) Create a new instruction that adds an integer to the content of a register. You must use 8 bits at

maximum

Code	Instrucction	Description
ADDi iii,RX	101iiirx	Adds an integer (iii) to the content of the register rr