IZMIR KATIP CELEBI UNIVERSITY

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

EEE458 MICROELECTRONIC CIRCUITS

MID-TERM EXAM OF 2019-2020 SPRING

Design the following VHDL modules and upload your answers as single zipped files of all VHDL files until 23:59 on May 2nd, 2020.

1. **Design an 8-bit register (PC):**

* when its "Increment" input is activated, it adds 1 to its content.

* when its "Load" input is activated, it stores the data given from 8-bit input of "Address".

* it gives its content to the 8-bit output of "PcOutput" directly.

2. **Design an 8-bit memory address register (MAR):**

* when its "Load1" input is activated, it stores the data given from 8-bit input of "Data1".

* when its "Load2" input is activated, it stores the data given from 8-bit input of "Data2".

* it gives its content to the 8-bit output of "AddressOutput" directly.

3. **Design a 256-item 12-bit memory (M):**

* it can store 256 different 12-bit data inside.

* it accepts an 8-bit input of "MemoryAddress".

* its 12-bit "MemoryData" is bidirectional. it can give outputs and inputs as you wish.

* when its "Read" input is activated, it gives the corresponding memory data item to its 12-bit bi-directional output of "MemoryData". for example, if the "MemoryAdress" is 15, then the 16th 12-bit data from the memory will be transferred to the output of "MemoryData".

* when its "Write" input is activated, it stores its 12-bit bi-directional input of "MemoryData" into the corresponding memory data item. for example, if the "MemoryAdress" is 15, then the input of "MemoryData" will be transferred to the 16th 12-bit data from the memory item.

4. **Design a 4-bit operation register (OPR):**

* when its "Load" input is activated, it stores the data given from 4-bit input of "OperandInput".

* it gives its content to the 4-bit output of "OperandOutput" directly.

5. **Design a 12-bit general-purpose register (GPR):**

* when its "Load1" input is activated, it stores the data given from 8-bit input of "DataPC" to its bits of 7 downto 0.

* when its "Load2" input is activated, it stores the data given from 12-bit bi-directional input of "DataMEM".

* when its "Load3" input is activated, it stores the data given from 12-bit input of "DataFromALU".

* it gives its content (from bits 7 to 0) to the 8-bit output of "AddressOutput" directly.

* it gives its content (from bits 11 to 8) to the 4-bit output of "OperandOutput" directly.

* it gives its content to the 12-bit output of "DataToALU" directly.

* when its content is 0, it gives 1 to the output of "Z"; otherwise, it gives 0 to the output of "Z".

* when its "Increment" input is activated, it adds 1 to its content.

6. **Design a simple arithmetic logic unit (ALU):**

* it has a 12-bit register inside named "ACC".

* it has a 1-bit flag inside named "F".

* when its "Clear" input is activated, it stores 0 into the "ACC".

* when its "Complement" input is activated, it stores the complement of actual value of "ACC" into the "ACC".

* when its "Increment" input is activated, it adds 1 to the actual value of "ACC" and stores this new data into the "ACC".

* when its "Add" input is activated, it adds the value of 12-bit "DataInput" to the value of "ACC" and stores the result into the "ACC" again. if a carry is occurred in this addition, then "F" becomes 1 else it becomes 0.

* when its "ClearF" input is activated, it stores 0 into the "F".

* when its "ComplementF" input is activated, it stores the complement of "F" into the "F" again.

* when its "ROR" input is activated, then it rotates right the content of "ACC" through "F". for example, if F=0 and ACC=010101010101, this generates F=1 (because the right-most bit of ACC is 1) and ACC=001010101010 (the left-most bit is transferred from F).

* when its "ROL" input is activated, then it rotates left the content of "ACC" through "F". for example, if F=0 and ACC=010101010101, this generates F=0 (because the left-most bit of ACC is 0) and ACC=10101010100 (the right-most bit is transferred from F).

* it gives its content of "ACC" to the 12-bit output of "DataOutput" directly.

* it gives its content of "F" to the 1-bit output of "Flag" directly.

P.S. You should also implement all these with separate top modules and simulate them 😊

GOOD LUCK !!!