**FENERBAHÇE UNIVERSITY**

**COMP2007- LOGICAL SYSTEM DESIGN**

**THE PROJECT DELIVERY REPORT**

metin, logo, simge, sembol, ticari marka içeren bir resim

Açıklama otomatik olarak oluşturuldu

**FB – CPU RTL DESIGN**

**Prepared by:**

Feridun Tavşanlı

Kevser Kahraman

Meryem Bürküt

Yüsra Azra Demirel

Zeynep İnce

**Course Instructor:**

Assist. Prof. Vecdi Emre LEVENT

**Date: 03.01.2024**

***1 – INTRODUCTION***

* 1. **The Purpose of The Project**

The purpose of this project is to develop the FB-CPU, a processor that can execute 9 different operations, such as addition and subtraction, and various code snippets written in machine language will be written on the designed processor. By the project's conclusion, it will be observed how this processors’s registers, RAM, control unit and processing unit can work together and execute code snippets in machine language.

Keywords — FPGA, CPU.

1. ***– THE SYSTEM ARCHITECTURE***

CPU basically has 4 elements. Those are register, memory unit (RAM), Arithmetic Logic Unit (ALU), Control Unit (CU).

The functions of the units: memory unit (RAM) is used by the processor to store the data while the program execution. Registers can store, and manipulate data during the execution instructions and do this through flip-flop circuits. Arithmetic Logic Unit (ALU) is used for all the arithmetic and logical operations. Control Unit (CU) decodes the commands and sends the data to the ALU for decoding.

* 1. **The Tools Used**

Using similar hardware design languages such as Verilog and VHDL, Xilinx Vivado Design Suite generates a design file that can be set to the FPGA during the FB-CPU design process.

"FB-CPU Simulator", which visualizes the architecture of the FB-CPU and allows the data flow to be observed, helped us see how the test software works.

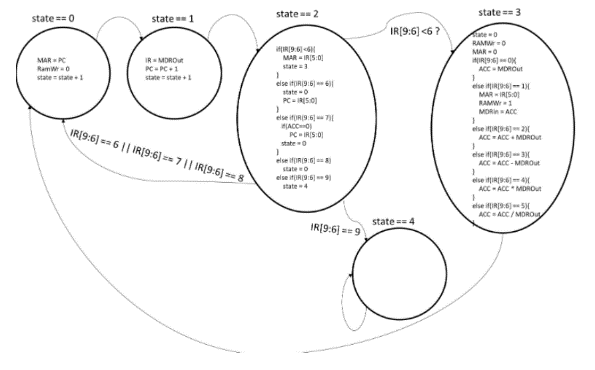
Von Neumann architecture was used in the FB-CPU RTL design.

* 1. **The Design**

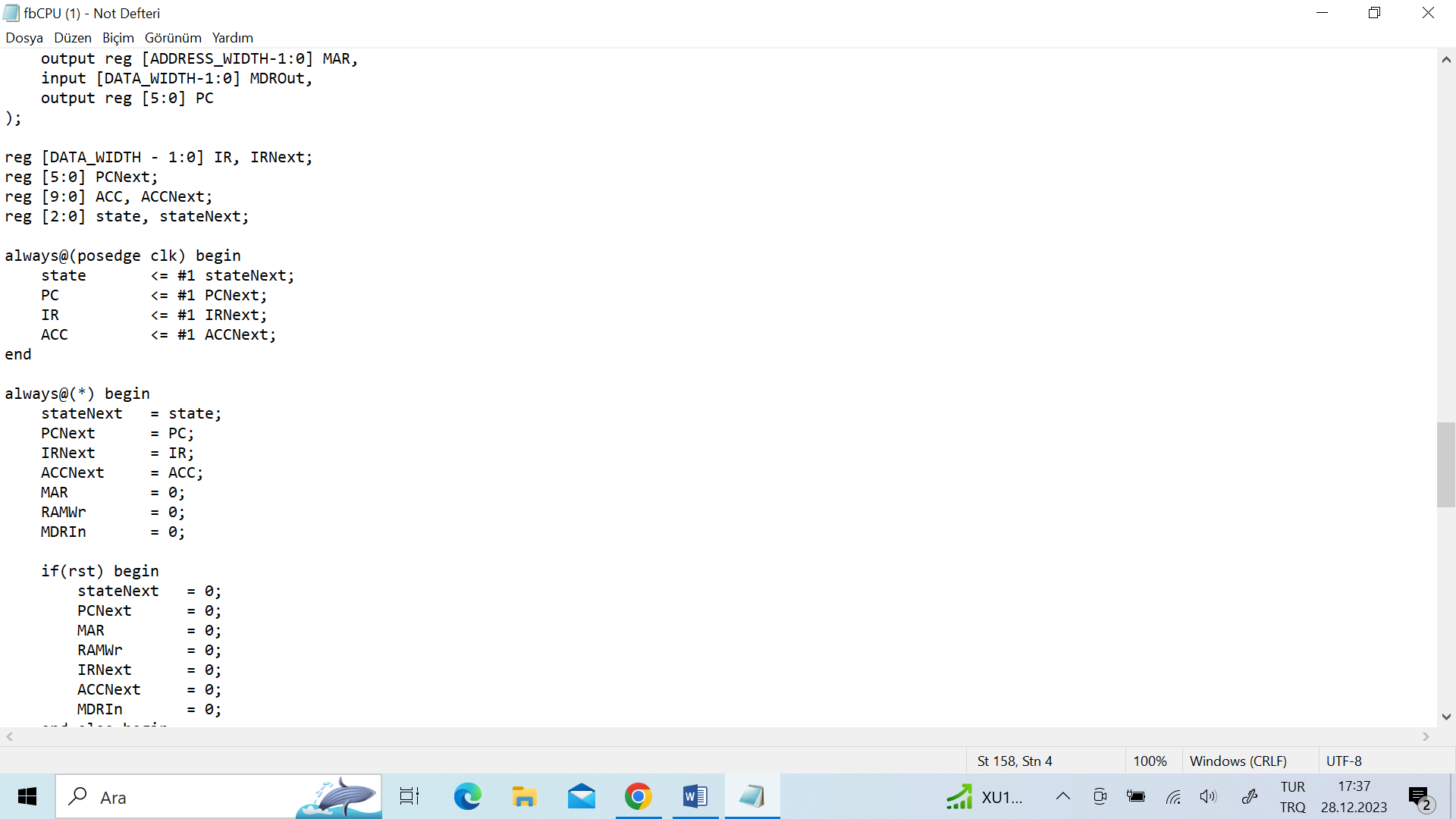
There are 4 registers in the code.

**State** is a variable that indicates the current state of a state machine or control unit.

The graph below shows the FB-CPU's state diagram. It outlines the tasks that the processor has to complete in detail.

****

**PC:** This is a register that stores the address of the microprocessor's next instruction in memory. The address of each instruction in memory is stored by the PC. As the commands are processed one after another, the PC displays the address of the next command.

**IR:** A register that temporarily stores instructions that a microprocessor is about to execute.

**ACC:** Accumulators are registers that store temporary results of arithmetic

and logical operations.

FB-CPU will be implemented with the state machine method. In other words, this processor will have a design that works in 5 different states, depending on the value of the register named state.

All other registers would operate based on the state register. In other words, the progress of the system depends on the status signal.

The memory signals connected to the input and output ports in the design are given below.

**MAR (6 Bit):** MAR is a register containing addresses used to access cells (addresses) in memory. This register is connected to the address input of RAM. RAM has 26 memory locations, so MAR is 6 bits. It is described as a 6-bit register, which means that MAR can represent 26, or 64 different memory addresses. Used to access cells in memory.

**MDRIn (10 Bit):** Memory data register In is an input data register used to write data to cells in memory. This register is used when writing data to RAM. Memory locations in RAM are 10 bits, so registers are also 10 bits. This means that MDRIn can display 210, or 1024, different data values. Transfers data to be written to cells in memory.

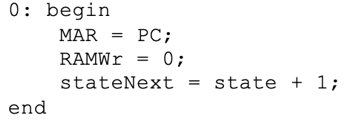
**RAMWr (1 Bit):** RAMWr is a control signal that controls whether writes are performed to cells in memory. If not 1, data is not written to RAM. This is specified as a 1-bit control signal and means that RAMWr can represent two different states (such as whether a write is being performed or not).

**MDROut (10 Bit):** Memory Data Register is output data registers that contain data read from memory. This is stated to be a 10-bit register. This means that MDROut can represent 210, or 1024, different data values. Carries data read from memory.

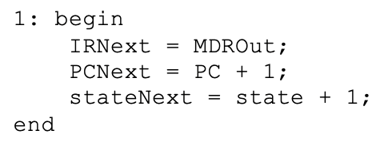
1. ***– THE DEVELOPED SOFTWARE***

***In state 0:***

1. MAR = PC;: The address within the Program Counter (PC) is transferred to the Memory Address Register (MAR). This sets the address for the next memory access.
2. RAMWr = 0;: The RAM write signal (RAMWr) is set to 0. In this state, no write operation to the memory occurs.
3. stateNext = state + 1;: The state is incremented to transition to the next state. In this case, the next state facilitates the execution of the next instruction.

* The Program Counter (PC) address is transferred to the Memory Address Register (MAR), no RAM write operation occurs, and the processor transitions to the next state. In this state, the processor becomes ready to read the instruction at the next memory address.

***In State 1:***

1. IRNext = MDROut;: The content of the Memory Data Register Output (MDROut), which holds the instruction read from memory, is assigned to the Instruction Register (IR). This step fetches the instruction from memory and loads it into the IR.
2. PCNext = PC + 1;: The Program Counter (PC) is incremented by 1, preparing it for the next instruction. This reflects the sequential execution of instructions in program memory.
3. stateNext = state + 1;: The state is incremented to transition to the next state. This prepares the processor for the next phase of the instruction execution cycle.

* In summary, state 1 is responsible for fetching the instruction from memory, loading it into the Instruction Register (IR), incrementing the Program Counter (PC) for the next instruction, and transitioning to the next state to continue the instruction execution cycle.

***In State 2:***

* if(IR[9:6] < 6) begin: The condition checks whether the top 4 bits of the instruction (IR[9:6]) are less than 6.

1. If this condition is true, i.e., if IR[9:6] is less than 6:
   * MAR = IR[5:0];: The Memory Address Register (MAR) is updated with the lower 6 bits of the instruction (IR[5:0]).
   * stateNext = 3;: Transition to state 3 is made. In this state, the processor will perform either a memory read or write operation.
2. If the condition is false:

* else if(IR[9:6] == 6) begin: Proceed to the next condition. In this case, if IR[9:6] is equal to 6:
  + stateNext = 0;: Transition to state 0 is made, meaning the execution of the next instruction.
  + PCNext = IR[5:0];: The Program Counter (PC) is updated with the lower 6 bits of IR. This allows the program to jump to a specific address.
* else if(IR[9:6] == 7) begin: If IR[9:6] is equal to 7:
  + if(ACC == 0) begin: If ACC (Accumulator) is equal to 0:
    - PCNext = IR[5:0];: The Program Counter is updated with the lower 6 bits of IR. This enables jumping to a specific address when ACC is 0.
  + metin, ekran görüntüsü, yazı tipi, doküman, belge içeren bir resim

    Açıklama otomatik olarak oluşturuldustateNext = 0;: Transition to state 0 is made, meaning the execution of the next instruction.
* else if(IR[9:6] == 8) begin: If IR[9:6] is equal to 8:
  + stateNext = 0;: Transition to state 0 is made, meaning the execution of the next instruction.
* else if(IR[9:6] == 9) begin: If IR[9:6] is equal to 9:
  + stateNext = 4;: Transition to state 4 is made. In this state, the processor prepares to jump to a specific address when a condition is met.
* These conditions allow for different control flows based on the values of specific bits in IR, enabling the processor to execute specific commands.

***In State 3:***

* stateNext = 0;: The state is set to 0, transitioning to the state where the next instruction will be fetched. This completes the current instruction execution cycle.
* RAMWr = 0;: The RAM write signal (RAMWr) is set to 0, indicating that no write operation to memory will occur in this state.
* MAR = 0;: The Memory Address Register (MAR) is cleared to 0, as it is not needed after this state.
* Conditional operations based on the value of IR[9:6]:

1. if(IR[9:6] == 0) begin: If the top 4 bits of the instruction are 0:

* ACCNext = MDROut; The content of the Memory Data Register Output (MDROut) is assigned to the Accumulator (ACC). This is typically a load operation.

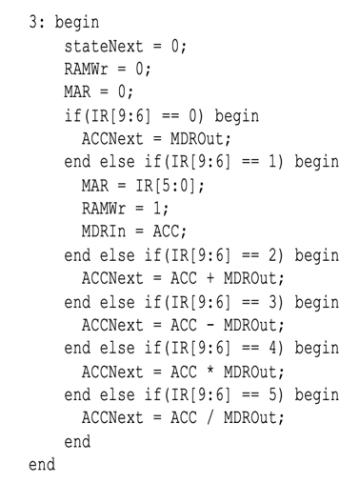
1. else if(IR[9:6] == 1) begin: If the top 4 bits of the instruction are 1:

* MAR = IR[5:0];: The Memory Address Register (MAR) is updated with the lower 6 bits of the instruction.
* RAMWr = 1;: The RAM write signal is set to 1, indicating a write operation to memory.
* MDRIn = ACC;: The content of the Accumulator (ACC) is written to memory at the address specified by MAR. This is typically a store operation.

1. else if(IR[9:6] == 2) begin: If the top 4 bits of the instruction are 2:

* ACCNext = ACC + MDROut;: The Accumulator is updated with the sum of its current value and the content of MDROut. This is typically an addition operation.

1. else if(IR[9:6] == 3) begin: If the top 4 bits of the instruction are 3:

* ACCNext = ACC - MDROut;: The Accumulator is updated with the difference between its current value and the content of MDROut. This is typically a subtraction operation.

1. else if(IR[9:6] == 4) begin: If the top 4 bits of the instruction are 4:

ACCNext = ACC \* MDROut;: The Accumulator is updated with the product of its current value and the content of MDROut. This is typically a multiplication operation.

1. else if(IR[9:6] == 5) begin: If the top 4 bits of the instruction are 5:

* ACCNext = ACC / MDROut;: The Accumulator is updated with the result of dividing its current value by the content of MDROut. This is typically a division operation.
* These operations represent the execution of arithmetic and data transfer instructions in the processor. The specific operation depends on the value of the top 4 bits of the instruction.

metin, yazı tipi, ekran görüntüsü, beyaz içeren bir resim

Açıklama otomatik olarak oluşturuldu

***In State 4:***

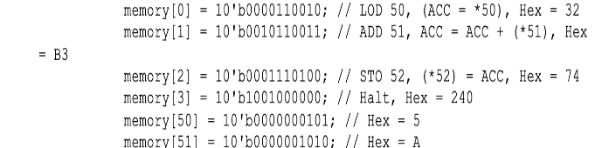
In case 4 nothing is checked, nothing is done it just spins on.

**Test Case 2:**

For FB-CPU, the number with the hexadecimal equivalent of 5 to the 50 addresses in memory is loaded into the ACC register with the LOD command. With the MUL command, the value of the 50th address in our ACC register is multiplied by the expression whose hexadecimal equivalent is A in address 51. With the STO command, the value of the ACC register (the product of addresses 50 and 51 is 32) is recorded in the 52nd address. The test was also stopped with the HLT command.

**Test Case 1:**

This test program loads values from specific memory addresses, performs arithmetic operations using the Accumulator, and stores results back into memory. It includes commands like load, add, and store. The program halts the processor after execution. Memory addresses 50 and 51 hold constants (5 and A in hexadecimal).

******

***metin, makbuz, yazı tipi, cebir içeren bir resim

Açıklama otomatik olarak oluşturuldu***

**Test Case 3:**

**Load Instruction (LOD):** Load the value from memory address 51 into the Accumulator (ACC). The loaded value is represented in hexadecimal as 33.

**Subtract Instruction (SUB):** Subtract the value from memory address 49 from the Accumulator (ACC). The result is stored back in the Accumulator. The hexadecimal representation of this command is F1.

**Jump if Zero Instruction (JMZ):** If the Accumulator is zero, jump to the instruction at memory address 10.

**Load Instruction (LOD):** Load the value from memory address 48 into the Accumulator (ACC).

**Add Instruction (ADD):** Add the value from memory address 50 to the Accumulator (ACC).

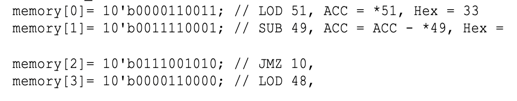
**Store Instruction (STO):** Store the Accumulator value into memory address 48.

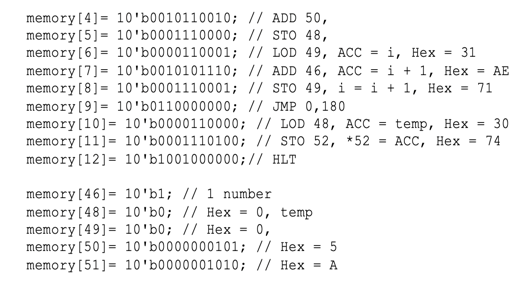
**Load Instruction (LOD):** Load the value from memory address 49 into the Accumulator (ACC). The hexadecimal representation of this command is 31.

**Add Instruction (ADD):** Add the value from memory address 46 to the Accumulator (ACC). The hexadecimal representation of this command is AE.

**Store Instruction (STO):** Store the Accumulator value into memory address 49.

**Jump Instruction (JMP):** Jump to the instruction at memory address 180.

**Load Instruction (LOD):** Load the value from memory address 48 into the Accumulator (ACC).

****Store Instruction (STO):** Store the Accumulator value into memory address 52.

**Halt Instruction (HLT):** Halt the processor, bringing the program to an end.

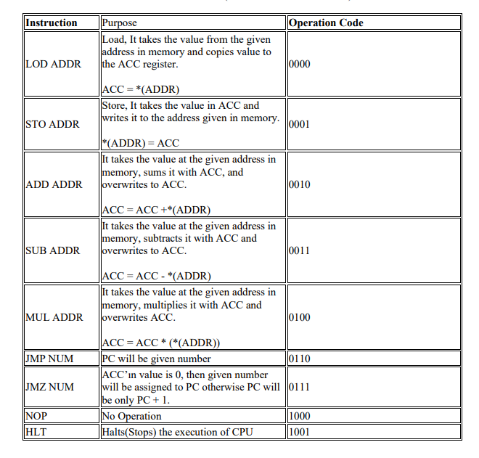
memory[46]= 10'b1;: Represents the value 1.

memory[48]= 10'b0;: Represents the value 0.

memory[49]= 10'b0;: Represents the value 0.

memory[50]= 10'b0000000101;: Represents the value 5.

memory[51]= 10'b0000001010;: Represents the value 10 in hexadecimal (A).



***4– RESULTS***

The operations supported by the developed processor are given in the FB-CPU ISA (Instruction Set Architecture) table.

Test software 1, test software 2 and test software 3 were tried one by one and examined through simulation on Vivado. Our algorithmic thinking ability improved and our experience in the Verilog language increased.

***5 - THE PROJECT TEAM***

Feridun TAVŞANLI was born on 01.10.2003 in Izmir/Konak. He graduated from Kartal Anatolian High School in 2021. He is currently studying for a bachelor's degree in Computer Engineering at Fenerbahçe University. His student number is 210304006. - feridun.tavsanli@stu.fbu.edu.tr

Kevser KAHRAMAN was born on 05.01.2001 in Çanakkale. She graduated from Kartal Yüksel İlhan Alanyalı Science High School in 2019. She is currently studying for a bachelor's degree in Computer Engineering at Fenerbahçe University. Her student number is 210304005. - kevser.kahraman@stu.fbu.edu.tr

Meryem BÜRKÜT was born on 31.03.2001. She graduated from Mustafa Saffet Anatolian High School in 2020. She is currently studying for a bachelor's degree in Computer Engineering at Fenerbahçe University. Her student number is 210304014. - [meryem.burkut@stu.fbu.edu.tr](mailto:meryem.burkut@stu.fbu.edu.tr)

Yüsra Azra DEMİREL was born on 30.01.2003. She graduated from Adnan Menderes Anatolian High School in 2021. She is currently studying for a bachelor's degree in Computer Engineering at Fenerbahçe University. Her student number is 210304022. - yusra.demirel@stu.fbu.edu.tr

Zeynep İNCE was born on 25.04.2002. She graduated from Cengizhan Anatolian High School in 2020. She is currently studying for a bachelor's degree in Computer Engineering at Fenerbahçe University. Her student number is 210304030. - [zeynep.ince@stu.fbu.edu.tr](mailto:zeynep.ince@stu.fbu.edu.tr)

***6 – REFERENCE FILES:***

1. ***YouTube link: https://www.youtube.com/watch?v=jGjxlb6vlOs***
2. ***GitHub link: https://github.com/zeynepiince/FB-CPU***

***7 - REFERENCES:***

1. [***https://www.learncomputerscienceonline.com/what-is-cpu/***](https://www.learncomputerscienceonline.com/what-is-cpu/)
2. ***http://www.levent.tc/files/courses/digital\_design/project/BLM201\_proje\_spesifikasyonlari.pdf***