******

Presented by:

Mohamed Adel Melad Abd\_Elhameed Youssef

Subject:

"Low Power Pipelined RISC-V 32\_bits

Processor with cache memory and FPU"

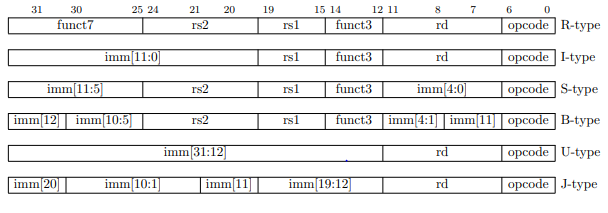
Under supervising of:

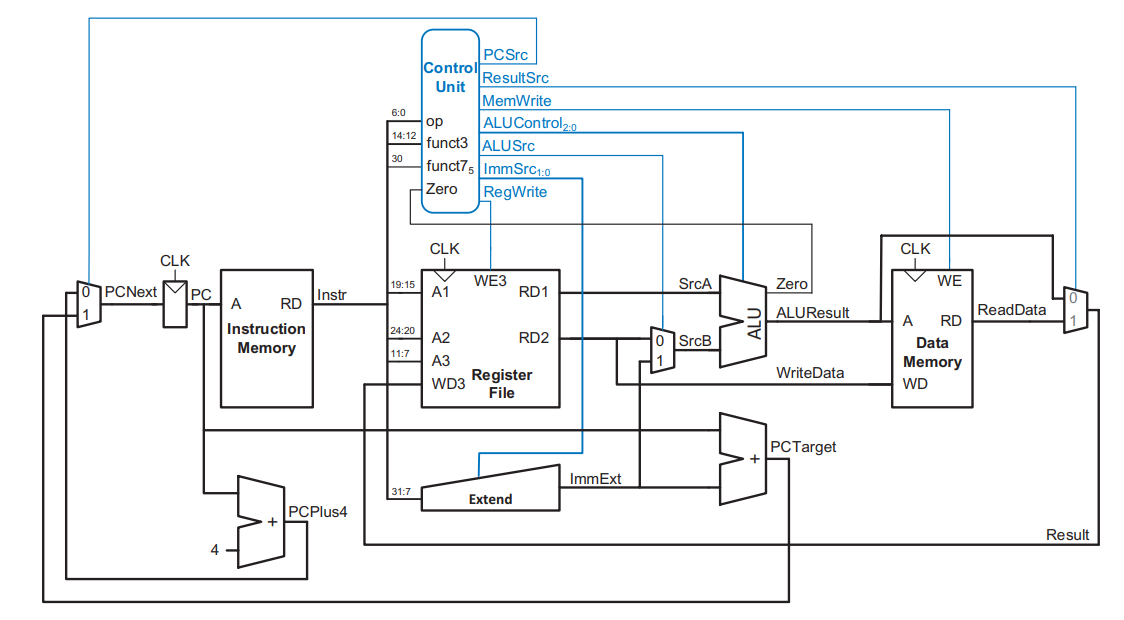
Eng: Ahmed

●To understand low power Pipelined RISC-V processor we need to know RISC\_V Single Cycle Processor very well.

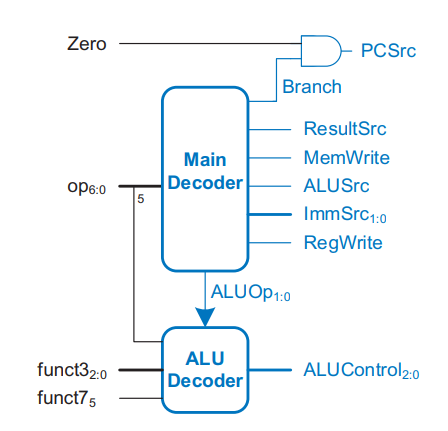
●So, I will explain RISC\_V Single Cycle Processor Description:

» The single-cycle data path, adding one piece at a time to the state elements from RISC\_V single cycle system Figure . The new connections are emphasized in black (or blue, for new control signals), whereas the hardware that has already been studied is shown in gray. The example instruction being executed is shown at the bottom of each figure. The program counter contains the address of the instruction to execute. The first step is to read this instruction from instruction memory. RISC\_V single cycle system Figure shows that the PC is simply connected to the address input of the instruction memory. The instruction memory reads out, or fetches, the 32-bit instruction, labeled Instr. In our sample program from RISC\_V single cycle system Figure, PC is 0x1000. (Note that this is a 32-bit processor, so PC is really 0x00001000, but we omit leading zeros to avoid cluttering the figure) .



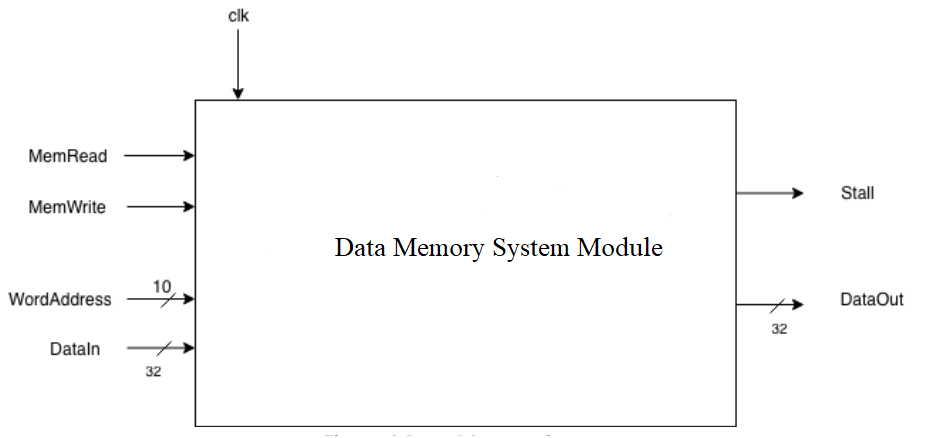


===========RISC\_V single cycle system Figure==========



===========RISC\_V single cycle control unit===========

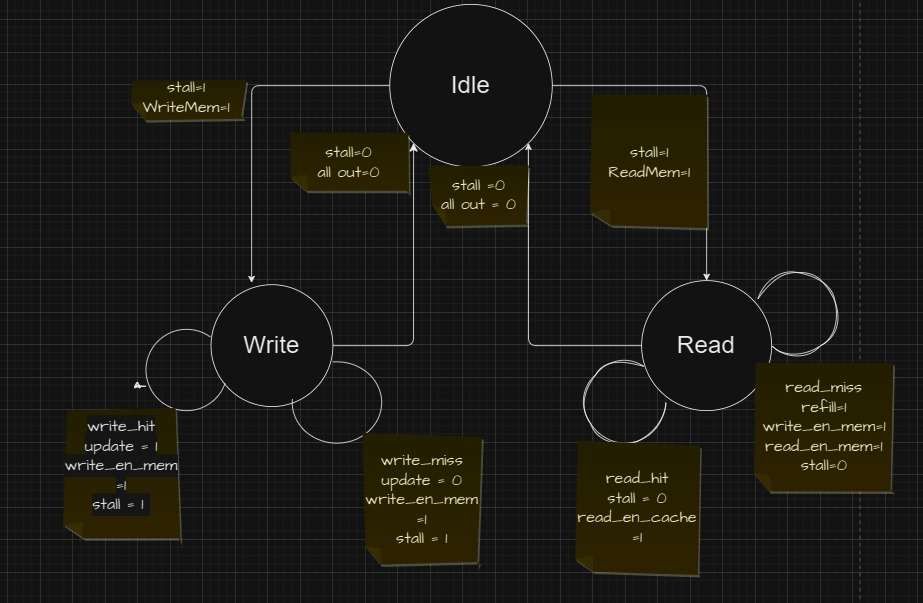
●We replace the data memory block in RISC\_V single cycle system with the data memory system module (RISC\_V single cycle system with cache memory) for optimizing the performance:



========= Top View Cache Memory Module========

»Then we use the stall signal as an enable signal for the program counter (PC). If the stall signal equal "**one**" the program counter (PC) will never get its next value. It will hold the processor from executing anything until it finish.

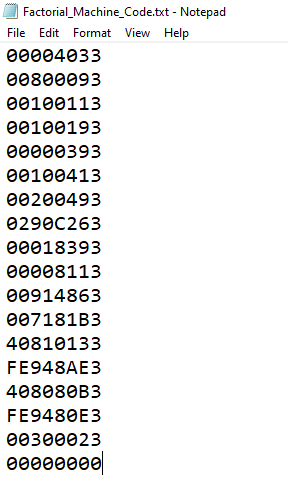
●Finite State Machine Controller Graph:



========== Finite State Machine Controller ==========

●What we need in testbench :

»Factorial\_Machine\_Code.txt :



●Simulation Result :

