 [Translated from Greek to English - www.onlinedoctranslator.com](https://www.onlinedoctranslator.com/en/?utm_source=onlinedoctranslator&utm_medium=docx&utm_campaign=attribution)

17/4/2018

# 4th Laboratory Exercise

**IMPLEMENTATION OF COMPLEX INSTRUCTIONS IN A MULTI-CYCLE PROCESSOR**

Group LAB31235515

|  |
| --- |
| *CHRISTOS ZISKAS 2014030191* |
| *ANTONIS ANTONAKAKIS 2014030160* |

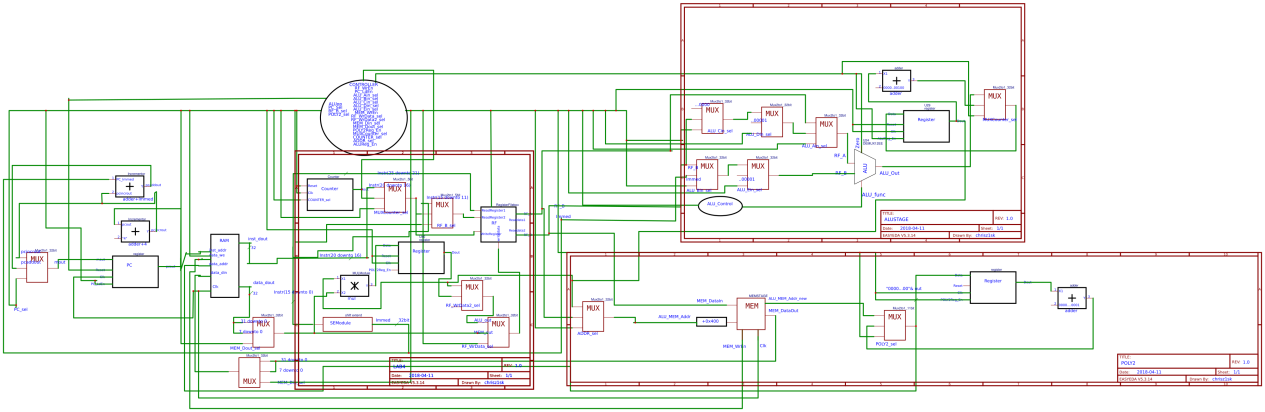
## Purpose of laboratory exercise

Expanding the functionality of the multi-cycle processor which refines the processor by giving it further dynamics. The command architecture is re-enriched, which entails additional control management for this purpose. At the same time, the strengthening of the design for the implementation of the new commands as well as for maintaining the functionality of the processor in terms of the execution of the already existing architecture.

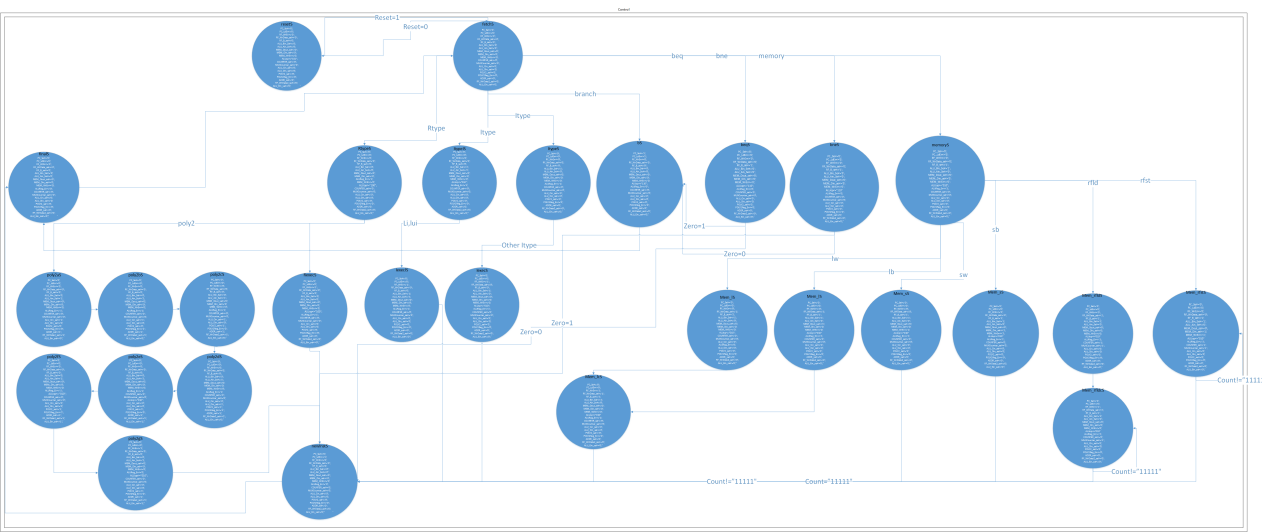
## Preparation

To complete the design of the multi-cycle processor, it is necessary to correct and evolve the control to accommodate the new set of instructions as new control signals are required to direct the datapath. Of course, the new control also needs a newer version of the data path that is consistent with the arrangement of the new commands. In addition, there is a small modification to the internal control unit to add the new calculations pushed by alu. The new command datapath concerns the input of multiplexes either for the input of the data to the appropriate addresses or as a correct output as well as the counter input that contribute to the execution of the rfst,rfld command. Without introducing a second alu module, a multiplication module -MULModule was created that occurs when the poly2 command is executed. Additional additions come with the presence of registers. The design of the command path as fsm is recommended below:

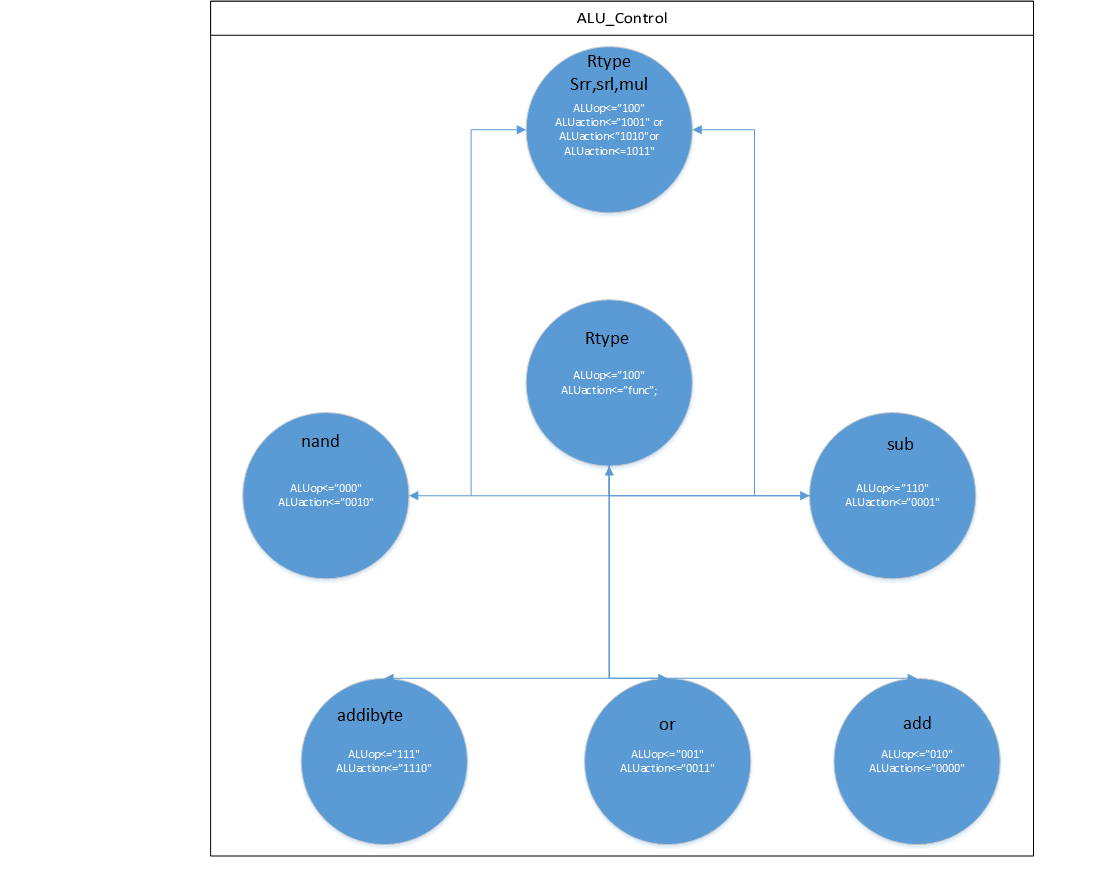
StepDatapath



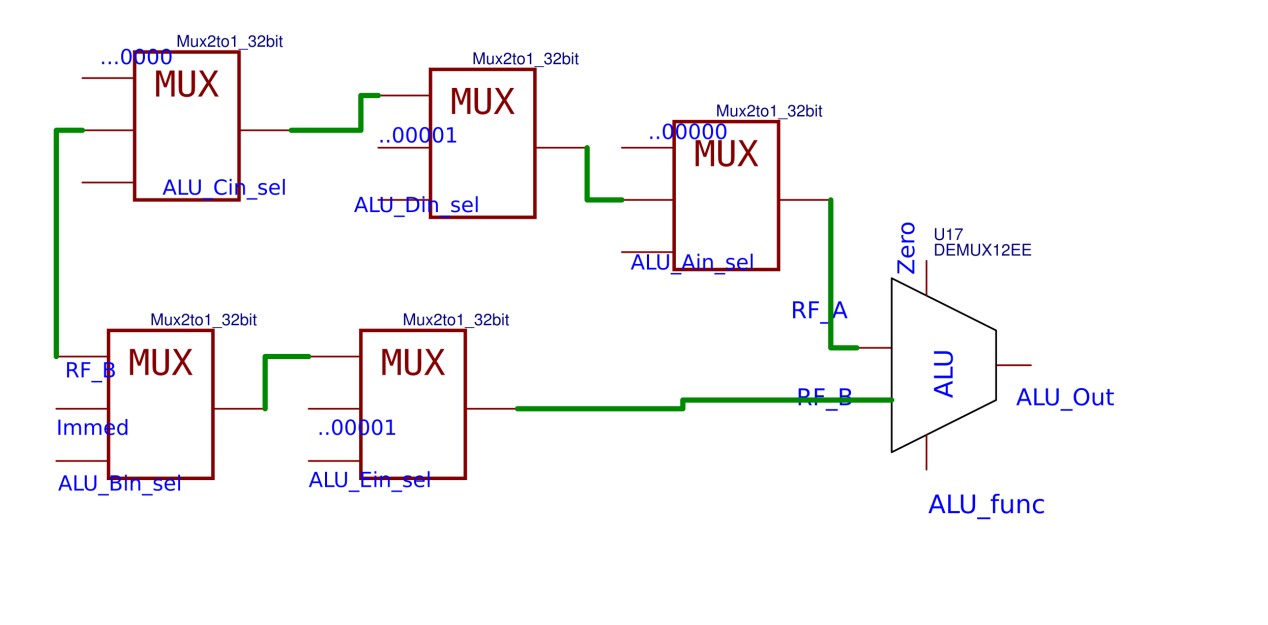
Control Unit (Control)



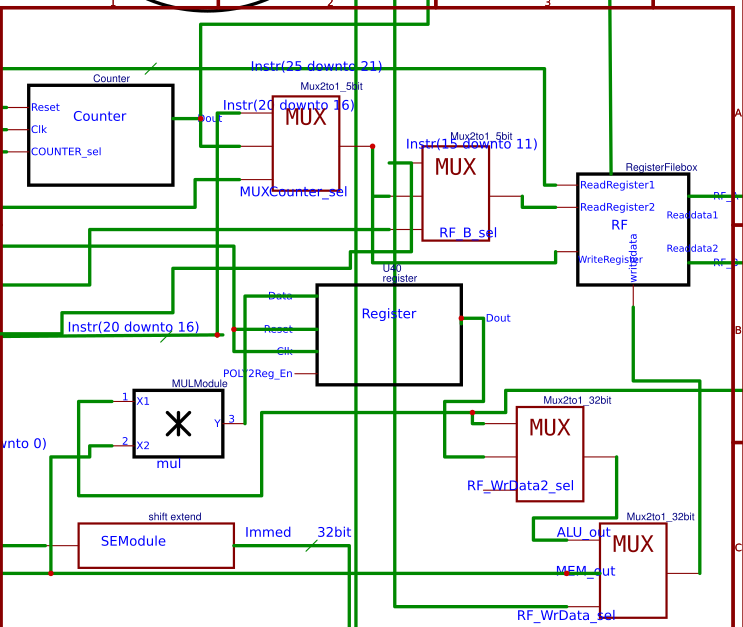
ALU Control Unit (ALU\_Control)



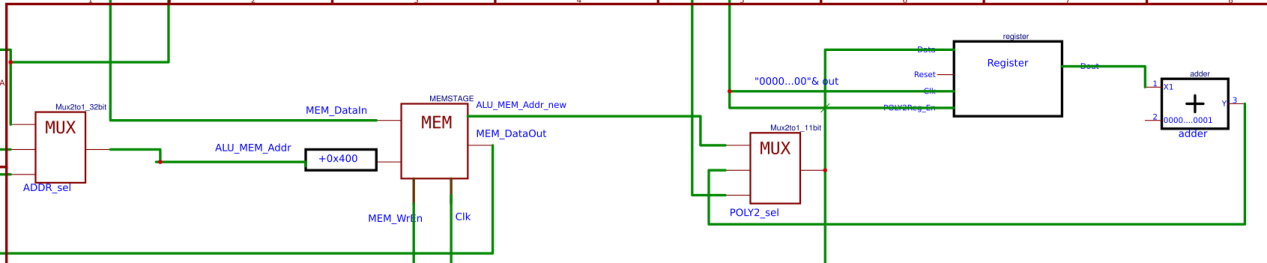
Calculation Unit (ALUstage)

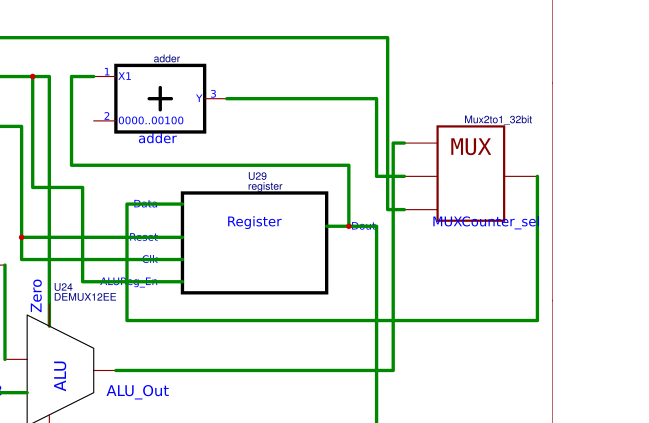


Decoding Unit (DECstage)



Amendments





## Description

The functionality of datapath and control on older architecture is not different in the new version. There are conversions due to the new elements performed by the processor.

The decoding stage (decstage) is upgraded by introducing a counter as well as additional multiplexers that contribute to the execution of the rfld,rfst instructions. The counter indicates which position in the register file is processed. As long as all positions of rf have not received a value from the command, the counter increases until it reaches the final position. A multiplexer is connected to the recording port, which accepts the counter at its input, so that during rfld execution, each of the positions in the register file is processed. Otherwise the multiplexer outputs the read address for destination[rd]. The output of the multiplexer is routed to the multiplexer which routes the second read port. The load of rf is extended in two cycles as it follows a similar logic as with the command lb,lw - duration is required to search for the requested address. In the second cycle, the enable of the counter is deactivated so that the expected positions are not exceeded. The extension is evolving with more functionality due to the introduction of new commands.

The computation stage (alustage) accepts transformations that help execute the poly2 command. Multiplexers are placed so that the appropriate data enters the alu. In the first phase, the second degree monomial must be found and the rest follow. So multiplexers are also used for this purpose. For theit is enough to place at the input of the alu multiplexers that accept as input the second output value of the rf while they will have as second input either 0 or 1 if the first or zero order monomial is necessary. For the zero-order monomial, a multiplexer is placed at the second input of the alu. By adding the alu unit also composes the function of multiplication for the requirements of the design. Therefore, a setting is also expected in the alu control unit for this purpose.

The datapath, in turn, has received variations. At the output of alu, an adder has been placed in the elements of a register that holds its output so that the desired addresses are processed when rfld,rfst is executed. The adder takes the result of the register and increments it by 4 - it is also an address and therefore advances to the next element. In any other command the result is passed directly to the multiplexer and continues its path.

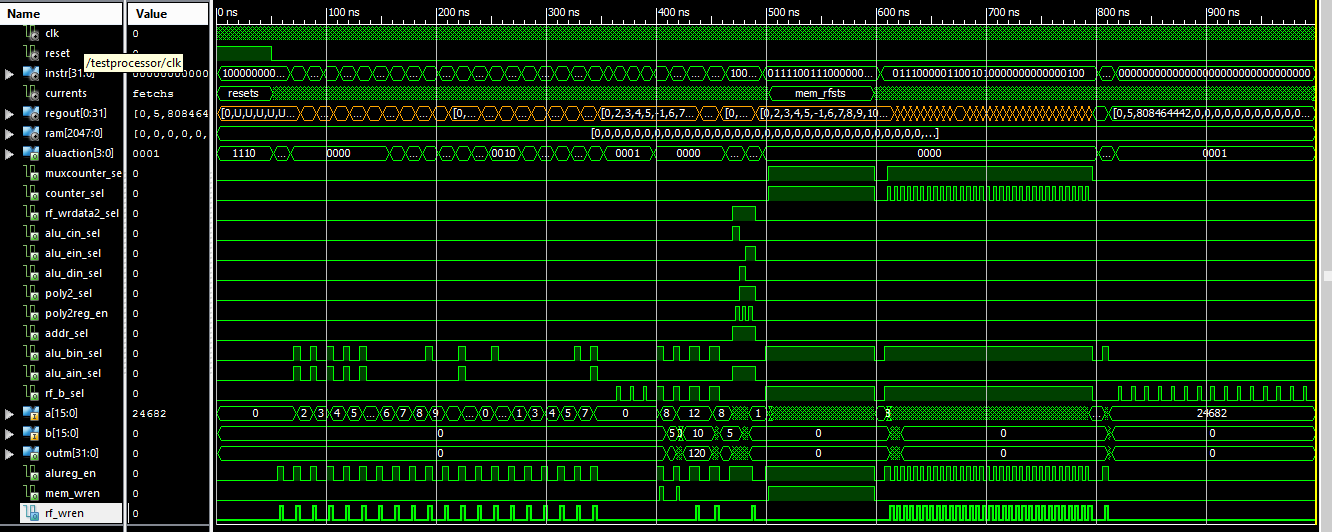
Regarding the address for the memory, a multiplexer has been placed that has as input the first value extracted from the register file to be input to the memory at the correct position MEM[RF[rs]] -refers to the command poly2-. Similarly an adder, register and multiplexer are placed so that there is a transition to the next element in memory from the given rs. Digging the results from memory handles a multiplier and a register. Multiplying the element of the register file with the result revealed from memory. The result remains in a register. With the new data entry, the register, keeping its value, is renewed by adding the new value to be entered. For this reason, the enable of the register remains active only for the moment of entry, otherwise there would be a change in the data. So here the sums of the monomials remain and the final result is stored in the register file. At this point the register is multiplexed and its output is connected to the rf register multiplexer.

The control is enriched with more states so that the signals respond to the system commands. The MMX\_addi\_byte instruction is only affected by the extension object as there were no variations due to this instruction in the datapath. The poly2 is executed in 3 phases of 2 cycles so a total of 6 cycles to search and register the correct results and in the 7th cycle the write is fulfilled. The rfst is executed as long as the counter value has not reached 11111. So 31 iterations. In each iteration a new output is written to the ram. The rfld is executed with the same logic only it needs a cycle to search the elements from the memory and with appropriate access to find the 31 consecutive positions. In that cycle the counter is deactivated and the action is set to register in the rf.

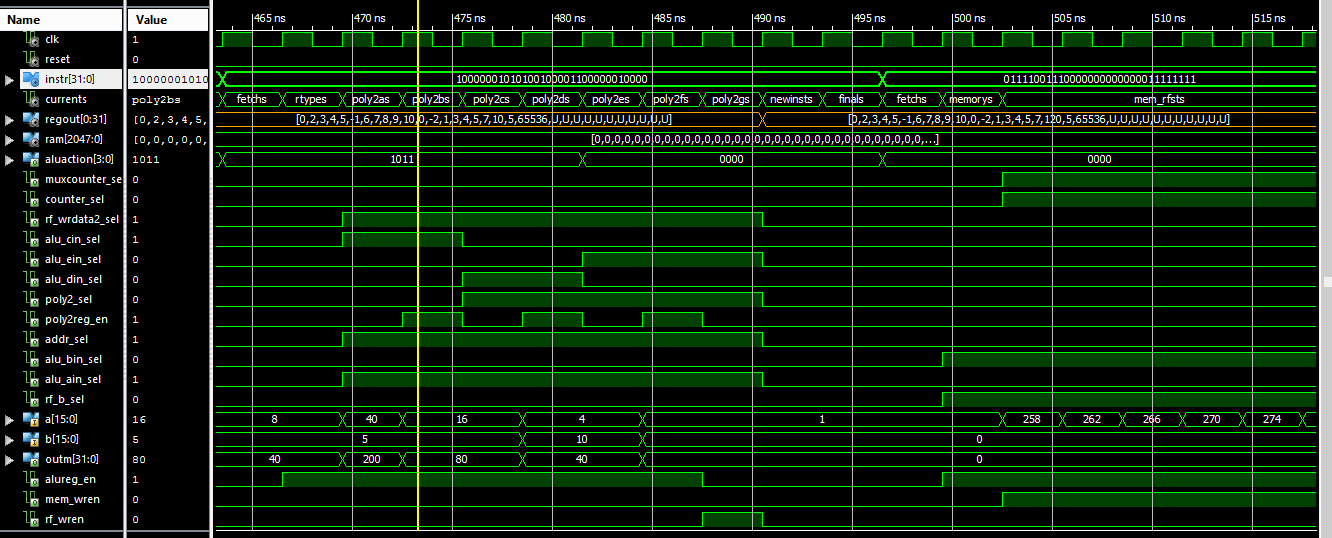
Essentially the new signals involved in the design perform only one function for the success of the circuit but are just as important as the rest.

## Waveforms-Simulation

The waveforms of the multi-cycle compound instruction processor are shown.



By analyzing the position of the processor during the execution of the commands - adding the new commands to the already existing rom file - the active signals are examined in each state.

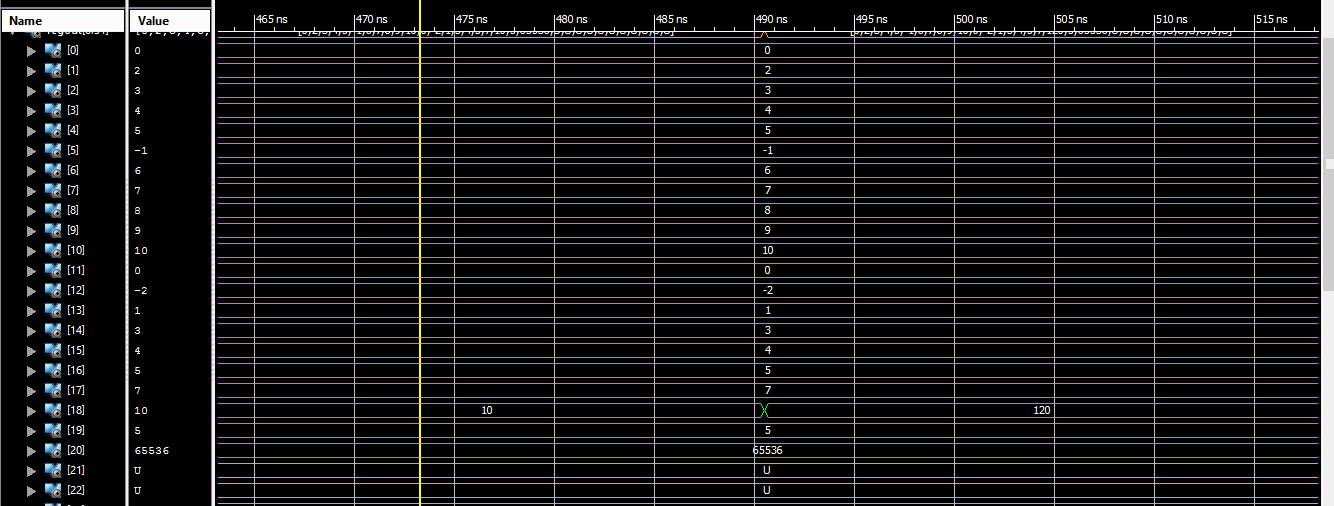


poly2 command --10000001010100100001100000010000

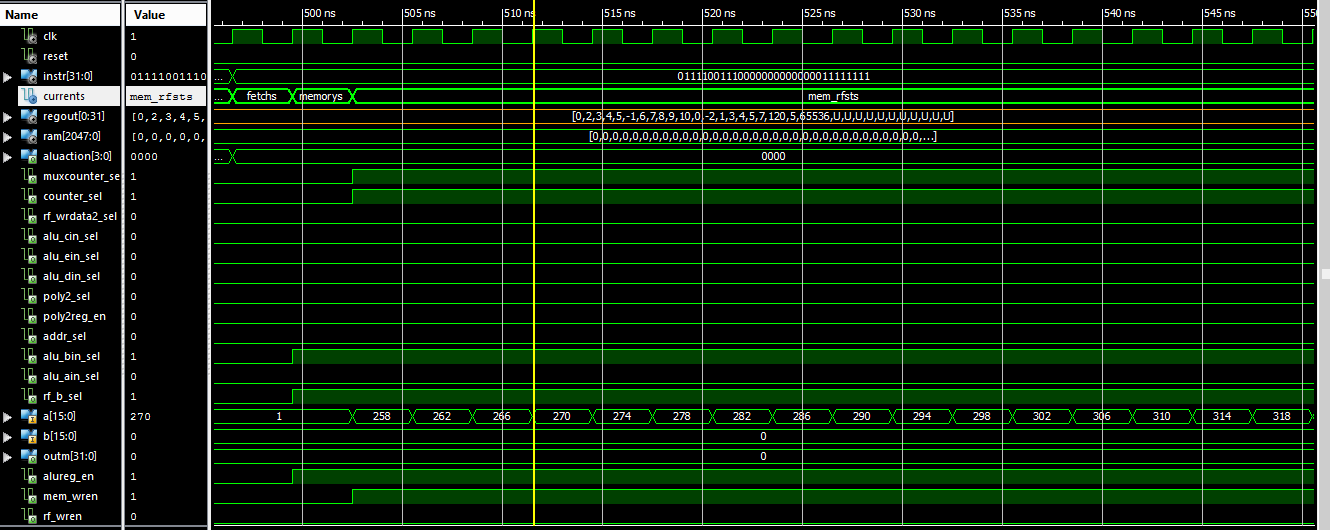
For the calculationalu\_cin\_sel & alu\_ain\_sel are activated so that the second output of the rf goes to the first and second input to the alu. Then their multiplication is performed and stored in the register. After the correct ram address is found, the multiplication is doneand the result is stored in a new register-the poly2regen flag is raised. To find the location RF[rs] in the ram a multiplexer is installed with inputs the first output of rf and the output of the register to the output of alu. In a second stage the command must executeso alu\_din\_sel & alu\_ain\_sel are raised and then moving to the next position inthe deed is done and then stored in the register. In the third stage, the function of alu is differentiated in addition, so the inputs of alu become zero and one respectively - alu\_ein\_sel is activated. Multiplying byis stored in the register . In the next cycle it composes an entry in rf.

The result:



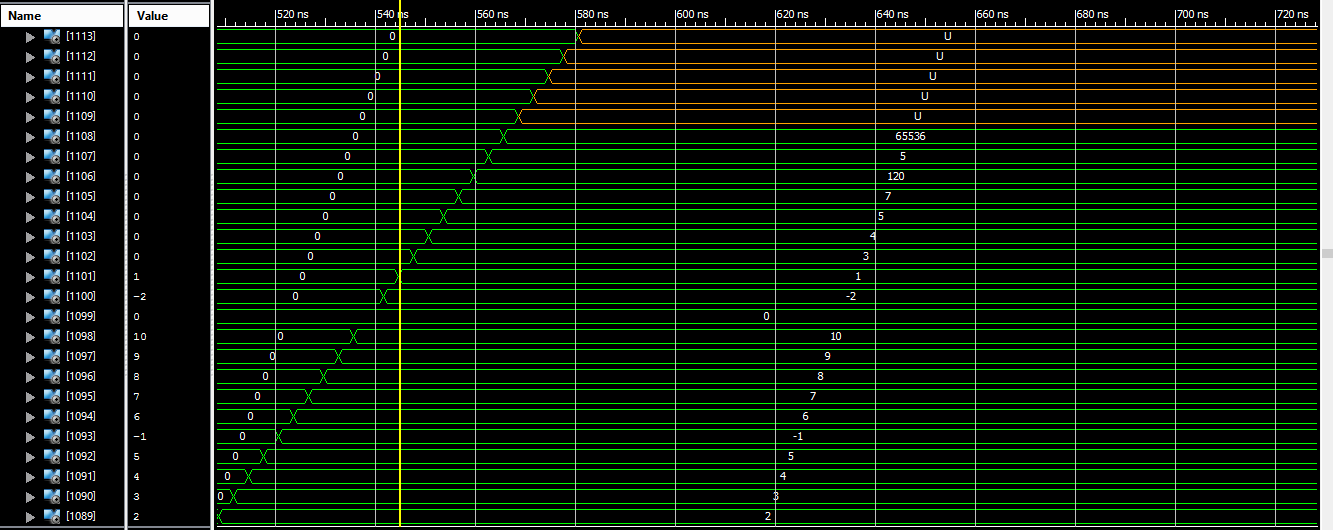
Examining the multiplication element -positions 1026 with entry 5 and 1027 with 10 stores element 120 in position 18.



rfst command -- 01111001110000000000000011111111

It takes one cycle to assign values ​​to ram from the register file. The counter is activated as well as the multiplexer which drives its value to the read port. Entries are made until all 32 register values ​​are processed and stored in consecutive ram locations. The address to be stored in the alu register is stored and for each increment of the counter, the address for the ram-addition is increased by 1. To complete the process, 32 repetitions of 1 cycle are required. Indications result in accurate results.

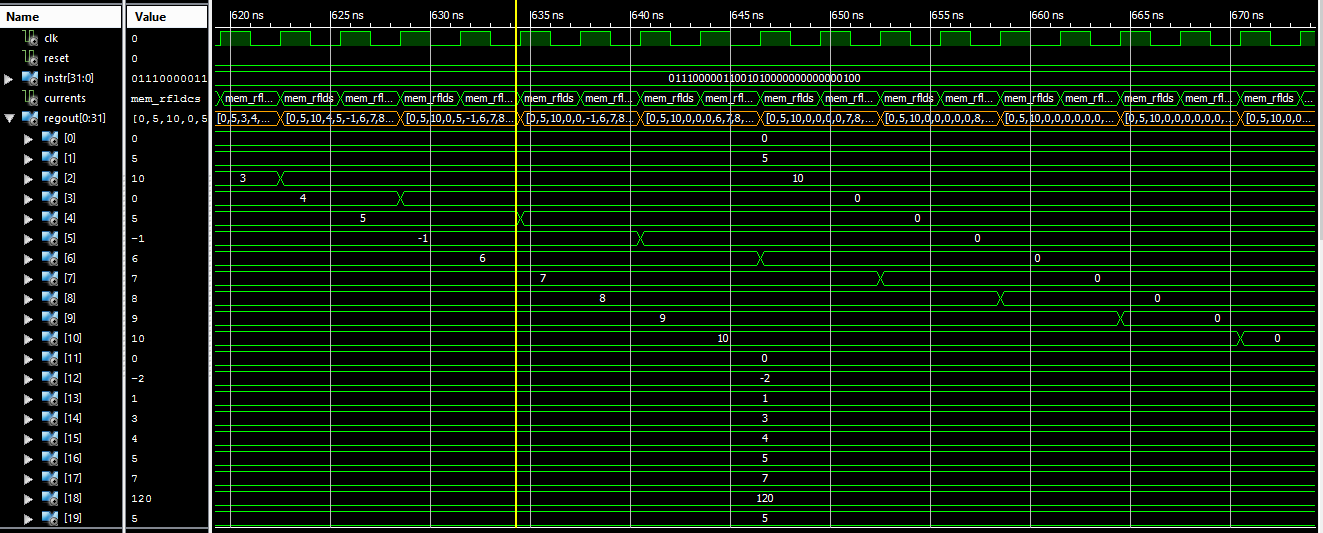
 



rfld command -- 01110000011001010000000000000100

Two cycles are required to assign values ​​to rf from the ram file. The counter is enabled as well as the multiplexer that drives its value to the write port. Writes are made until 32 consecutive ram values ​​have been processed and stored in all positions in rf. The address to take the values ​​from ram is stored in the first cycle while in the second the result of that location is stored in the file.



The register file accepts the values ​​accumulated in the ram locations

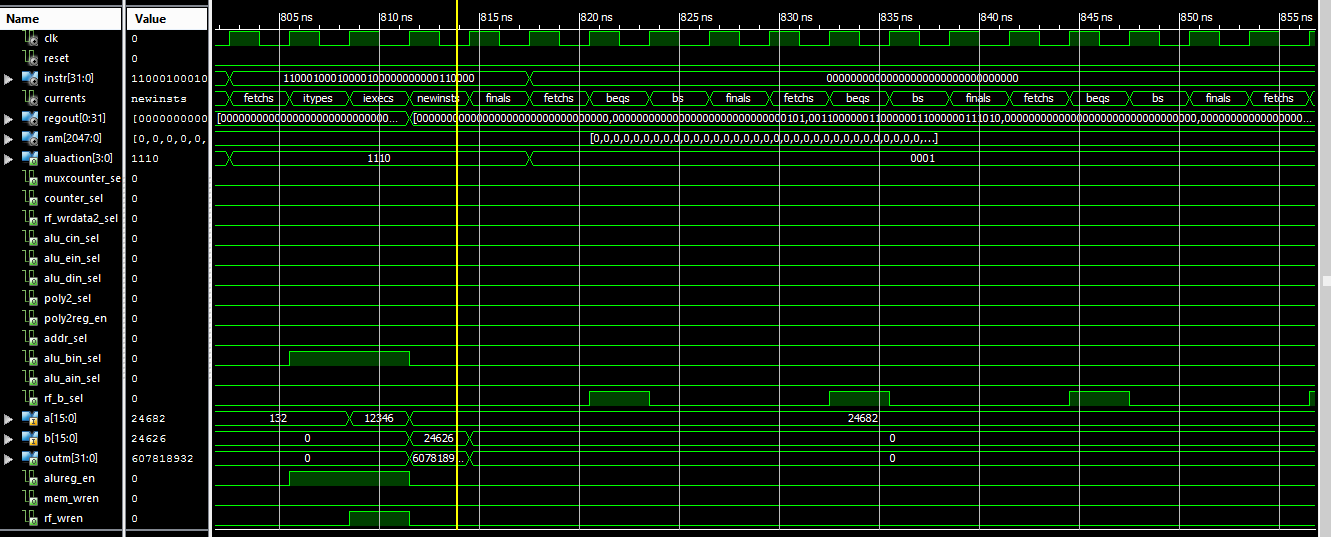
 

mmx\_addi\_byte command --11000100010000100000000000110000

The command is no different from the Immediate commands. Design modification for this command is covered by the extension module and the result is saved. In the extension object, the last byte of the immidiate in each of the four bytes is extracted piecewise. Different from li,lui commands - alu\_bin\_sel is enabled. alu manages with a separate code to extract the output of the addition of each byte between immidiate and RF[rs]. The writing is done in position 2 of the RF and the same position is set for reading.

|  |  |  |  |
| --- | --- | --- | --- |
|  | RF[rs] | Immediately | Result |
| 7 down to 0 | 00001010 | 00110000 | 00111010 |
| 15 down to 8 | 00000000 | 00110000 | 00110000 |
| 23 down to 16 | 00000000 | 00110000 | 00110000 |
| 31 down to 24 | 00000000 | 00110000 | 00110000 |

So: 00110000001100000011000000111010 which is also the desired



## Conclusions/Problems

The laboratory exercise leads to important conclusions since the design of a complex multi-cycle processor with elaborate operation is unified. The conclusion on this type of design is summed up in the complexity of the operations that a processor can accomplish and therefore take effect according to new commands, new operations through the existing logic.