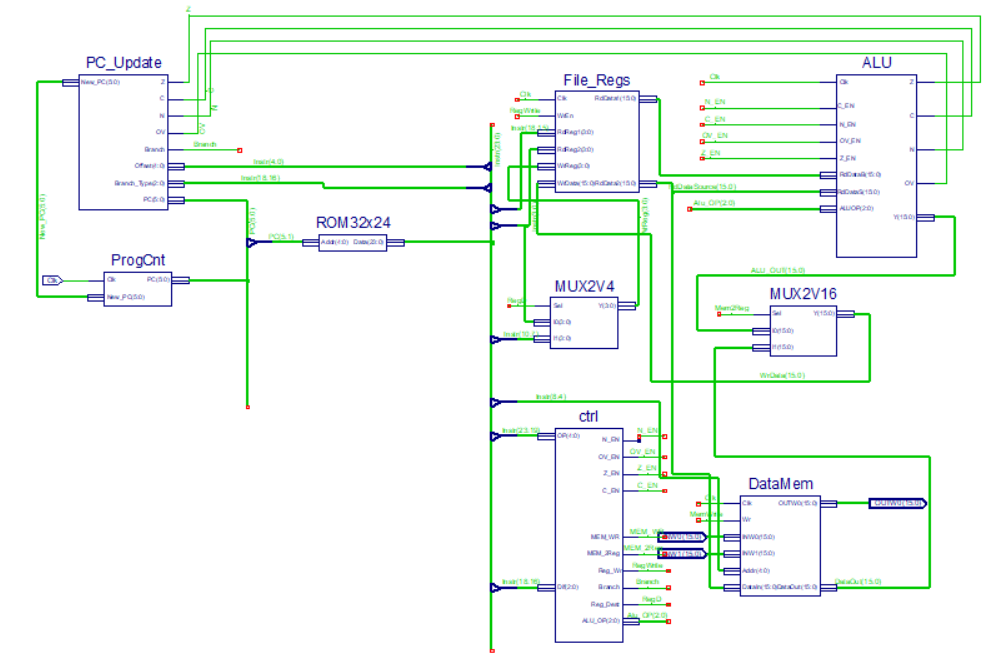
PIC24 – Project

For this project we had to design a processor that will execute a subset of the PIC24E instruction set.

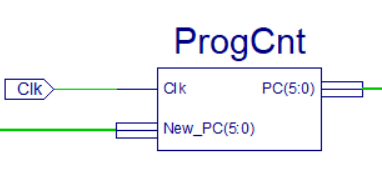
The block diagram is as follows:



The PIC24 is composed of 9 blocks which we will describe in order:

**The ProgCnt**

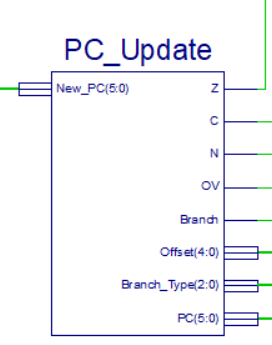
The block diagram for it:

****

The Program Counter is a register in the processor that contains the address of the instruction that is being executed at the current time. As each instruction gets fetched, the Program Counter increases its value by 1. Here, we receive the updated value of the PC on the rising edge of the clock signal.

**The PC\_Update**

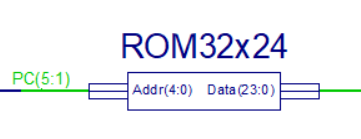
The block diagram for it:



The PC\_Update, as said in its name, updates the value of the ProgramCounter. If a branch type instruction is encountered and the conditions imposed for updating the PC are met (flags=1), then the PC will receive PC+2 (jump to next instruction) + the offset. If no branch instructions are met or the conditions imposed for updating the PC are not met then the PC will receive PC + 2. The offset is found in the first 5 bits of the instruction.

**The ROM32x24**

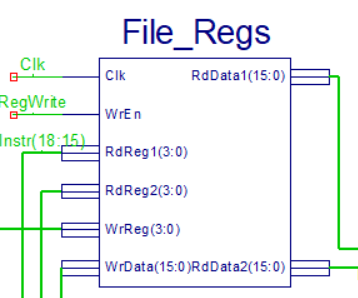
The block diagram for it:



This block represents the memory of the program with a capacity of 32 instructions each on 24 bits. For this reason, it has 5 inputs and 24 outputs, the output being the instruction selected by the ProgramCounter.

**File\_Regs**

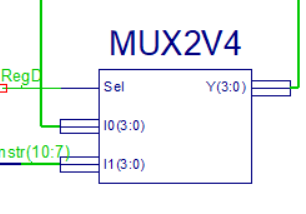
Block diagram:



The processor register is a quickly accessible location available to a computer's processor. Into the register we load data from a larger memory in order for it to be used for arithmetic operations and manipulated or tested by machine instructions. The general register block contains 16 registers(W0-W15), each on 16 bits. This block has 2 read ports and a write port, because there are instructions that require reading 2 registers and writing 1 register simultaneously. Manipulated data can always be stored back into the main memory.

**The MUX2v4**

Block diagram:

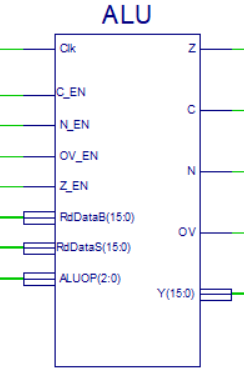


I needed to use this MUX because not all instructions have the same format, for some of them the destination register address is in bits 3:0 and then for others in bits 10:7.

This Mux has as output, depending on the RegDest signal, the bits used to select the destination register for the current instruction.

**The ALU**

Block diagram:



This block is responsible for all the arithmetic and logic operations of the processor.

Our processor, Pic24 is equipped with flags that are implemented in ALU.

A flag is a bit that provides information about a certain property of the result of the operation performed by ALU. The ALU\_OP decides what operation to be performed by ALU with the operands received at input.

The values of the signals are determined in ALU as follows:

The flag N is activated when the result is negative, which means when bit 15 has the value 1.

The Z flag is activated when the result of an operation is 0.

The Flag C is activated when after the operation we have a carry, which happens when adding or subtracting 2 15-bit operands results in a 16-bit result.

The OV flag is set whenever the result of a signed number operation is too large, causing the high-order bit to overflow into the sign bit.

In the ALU are also implemented the specific instructions of the program:

(FF1l) - finds the first occurrence of a ‘1’ bit starting from the most significant bit of Ws and working towards the least significant bit of the word operand. Then the bit number result is zero-extended to 16 bits and placed in Wnd.

(SL) - shifts the contents of the source register Ws one bit to the left and places

the result in the destination register Wd. The Most Significant bit of Ws is

shifted into the Carry bit of the STATUS register, and ‘0’ is shifted into the

Least Significant bit of Wd.

(COM) - computes the 1’s complement of the contents of the file register and place

the result in the destination register.

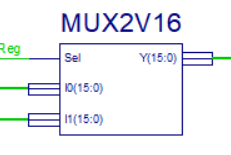
(IOR #lit5) - computes the logical inclusive OR operation of the contents of the base

register Wb and the 5-bit literal operand and place the result in the

destination register Wd.

**The MUX 2V16**

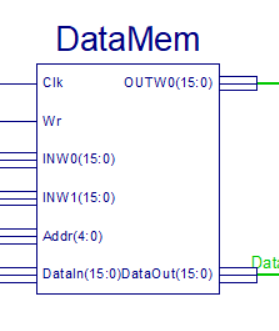
Block diagram:



This multiplexer decides with the help of the Mem2Reg signal whether the data written in the register will come from ALU or from memory (with instruction MOV f, Wnd).

**DataMem**

Block diagram:



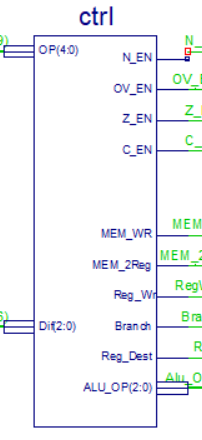
This is the RAM block that contains a vector of 16 elements of 16 bits each.

Reading is done from addresses 1020h(INW0) and 1022h (INW1) and writing is done at address 1024h (OUTW0).

The DataOut output will receive either INW0, INW1 or MemData which represents the value selected from the 16-element vector using the number resulting from converting the Addr vector to an integer (3 downto 0). The A4-A0 addresses generated by the processor connect the Addr(4:0) pins of the memory block.

**The Control Block**

The block diagram:



This block operates with decoding the OPCODE of the instructions and with setting the control signals as well.

I had to also implement the Dif because two instructions (IOR and IOR Wb,#lit5,Wd) had the same OPCODE, so I chose the next 3 bits for the difference to know which is which.

**The control signals:**

AluOP - represents the operation executed by the ALU block. It is set depending on the OPCODE in each instruction

MemWr - it is used to condition writing in memory

Mem2Reg - when this signal is active, through the MOV instruction f, Wnd, a 16-bit word is transmitted from the memory to the register

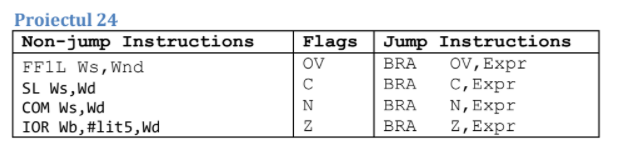
Branch - represents the jump operation conditioned or unconditioned by flags. With the help of this signal the jump in the block PC\_UPDATE is made

RegWr - it is used to condition the writing in the register

RegDest - is set using the OPCODE resulting from each instruction and represents the selection input from the MUX2v4 multiplexer, in which the input of the destination bits is selected

RegRead - is set using the opcode resulting from each instruction and represents the selection input from the MUX2v4 multiplexer in which the bit input indicating the address of the basic register is selected

I had been assigned the project Number 24, so my specific instructions are the following:



The truth table for the instructions is the following:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Encoding | OP | Flag N | Flag OV | Flag Z | Flag C | ALU OP | MEM Wr | MEM 2Reg | Reg Wr | Branch | Reg Dest |  |
| ADD | 01000 | 1 | 1 | 1 | 1 | 000 | 0 | 0 | 1 | 0 | 1 |  |
| SUB | 01010 | 1 | 1 | 1 | 1 | 001 | 0 | 0 | 1 | 0 | 1 |  |
| AND | 01100 | 1 | 0 | 1 | 0 | 011 | 0 | 0 | 1 | 0 | 1 |  |
| IOR | 01110 | 1 | 0 | 1 | 0 | 010 | 0 | 0 | 1 | 0 | 1 |  |
| MOV f,Wnd | 10000 | 0 | 0 | 0 | 0 | - | 0 | 1 | 1 | 0 | 0 |  |
| MOV Wns,f | 10001 | 0 | 0 | 0 | 0 | - | 1 | 0 | 0 | 0 | 0 |  |
| BRA expr | 00110 | 0 | 0 | 0 | 0 | - | 0 | 0 | 0 | 1 | - |  |
| FF1L Ws,Wnd | 11001 | 0 | 0 | 0 | 1 | 100 | 0 | 0 | 1 | 0 | 1 |  |
| SL Ws, Wd | 11010 | 1 | 0 | 1 | 1 | 101 | 0 | 0 | 1 | 0 | 1 |  |
| COM Ws, Wd | 11101 | 1 | 0 | 1 | 0 | 110 | 0 | 0 | 1 | 0 | 1 |  |
| IOR Wb, #lit5, Wd | 01110 | 1 | 0 | 1 | 0 | 111 | 0 | 0 | 1 | 0 | 1 |  |
| BRA OV,Expr | 00110 | 0 | 0 | 0 | 0 | - | 0 | 0 | 0 | 1 | - |  |
| BRA C,Expr | 00110 | 0 | 0 | 0 | 0 | - | 0 | 0 | 0 | 1 | - |  |
| BRA N,Expr | 00110 | 0 | 0 | 0 | 0 | - | 0 | 0 | 0 | 1 | - |  |
| BRA Z,Expr | 00110 | 0 | 0 | 0 | 0 | - | 0 | 0 | 0 | 1 | - |  |