**Buzduga Ionut Gabriel-CEN3.1B**

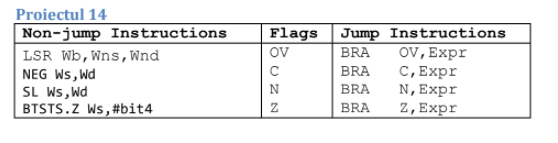
**Project 14**

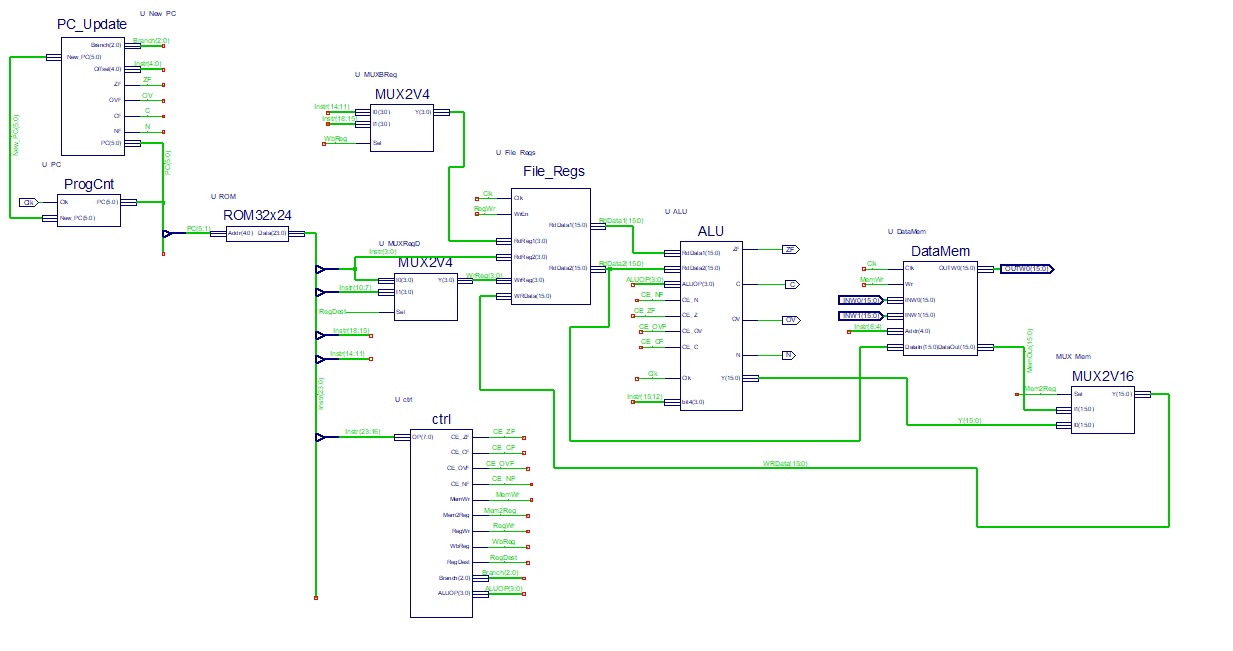
**Project number:14**

**Project requirements:**

-Implement a PIC24E processor with all the common and specific Instructions, as well as the Flags and Jump Instructions

**Project 14 Specific Instructions:**

****

**The Block Diagram of the Processor is**

**The Role of The blocks ands signals of the processor:**

**PC\_Update(which eiter increments The program counter(ProgCnt) by 2 or by 2+Offset when encountering a branch instruction)**

Signals:

-New\_Pc-which holds the value of the new program counter

-PC-the current program counter

-Offset-its value is needed for the branch instructions

-Branch-a 3 bit signal for the value of each different branch instruction

-ZF,OVF,CF,NF which are flag signals received from ALU And ctrl blocks

**Rom32x24(it is the program memory which holds 32 instructions of 24 btis each)**

Signals:

Data-which has the value of which instructions to execute

Addr-where we get the address of the instruction selected

**ProgCnt(gets the New\_Pc value from Pc\_Update when the clk signal is on the rising edge(logical 1))**

Signals:

-Clk-is set to oscillate between high and low in the testbench

-PC-the current program counter

-New\_PC -the new value of the program counter

**Ctrl** **–(This is where the Flags and Registers get their values;also this is where the instructions are ordered and signaled to the ALU block using ALUOP signal)**

Signals:

CE\_ZF,CE\_CF,CE\_OVF,CE\_NF-signals used for the zero,carry,overflow and negative bit signaling to the ALU block

BaseReg –Is the register used for the specific instruction LSR Wb,Wns,Wnd to choose either the 14:11 bits when the OPCODE matches that of the instruction, or the 18:15 bits otherwise; It is also tight to the Base register Mux which makes that selection

RegDest – signal which indicates if the destination register is either on the 3:0 bits or on the 10:7 bits;it is tight to the Reg Dest MUX which makes the selection

MemWr – it signals if the instruction uses memory write

Mem2Reg – it writes the data from ALU to the register;but if the signal has the value 1, then the data will be written from the memory to the register

RegWr – signal which indicates writing in a register

.

ALUOP(3:0) – it determines which operation to execute in the ALU block

**ALU(it implements the common and specific instructions as well as getting the value for the flag signals)**

Signals:

CE\_Z,CE\_N,CE\_OV,CE\_C and Z,N,OV,C-are used to indicate the flags signals.

RdData1(15:0)– input operand for the base register

RdData2(15:0)– input operand for the source register

Y – 16 bit output signal

Clk – clock signal

ALUOP(3:0)– selects which operation to execute in the ALU block

bit4-this is a signal used for the specific instruction(BTSTS.Z Ws,#bit 4) because we need to select the 15:12 bits because that is where the bit4 value is in the instruction which we can further use to implement the instruction.

**File\_Regs (this is where the registers are implemented;it has two registers for reading and one for writing each on 16 bits)**

Signals:

Clk – Clock signal

WrEn – validates the writing in the register

RdReg1(3:0) – bits for the base register

RdReg2(3:0) – bits for the source register

WrReg(3:0) – indicates the number of the 16 registers to write

WrData(15:0) – this is the data to write

RdData1(15:0) – holds the contents of register1

RdData2(15:0) – holds the contents of register2

**DataMem (this is the RAM memory which contains contains the addresses for reading and writing either in the register or in the memory)**

Signals:

Clk – Clock signal

INW0(15:0) and INW1(15:0)- addresses for reading

Wr- when this signal is high it writes DataIn to the memory

Addr(4:0) -bits for adress

DataIn(15:0)– value to be written

DataOut(15:0) – which gets either INW0 or INW1

OUTW0(15:0) – address for writing which it gets from DataIn

We have 2 MUX2V4:

The first one is U\_MUXRegD which is used to select between the mov f, wnd instruction and the other instructions

It selects bits 10:7 through input signal I1 for all the other instructions and bits 3:0 (trough I0)for mov f,wnd instruction.

Similarly U\_MUXBReg Is used for the LSR Wb,Wns,Wnd instruction to select between the 14:11(I0) to execute this instruction and 18:15(I1) to execute other instructions.

MUX2V16 – it decides through Mem2Reg if the ALU data comes from memory(I1) or from ALU(I0)

The truth tables used in this project:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Encoding** | 2222  3210 | 1111  9876 | 1111  5432 | 11  1098 | 7654 | 3210 | Flags |
| ADD | 0100 | 0www | wBqq | qddd | dppp | ssss | N, OV, Z, C |
| SUB | 0101 | 0www | wBqq | qddd | dppp | ssss | N, OV, Z, C |
| AND | 0110 | 0www | wBqq | qddd | dppp | ssss | N, -, Z, - |
| IOR | 0111 | 0www | wBqq | qddd | dppp | ssss | N, -, Z, - |
| MOV f, wnd | 1000 | 0fff | 0fff | 0fff | 0fff | dddd | none |
| MOV wns, f | 1000 | 1fff | ffff | ffff | ffff | ssss | none |
| BRA expr | 0011 | 0111 | nnnn | nnnn | nnnn | nnnn | none |
| LSR Wb,Wns,Wnd | 1101 | 1110 | 0www | wddd | d000 | ssss | N, Z |
| NEG Ws,Wd | 1110 | 1010 | 0Bqq | qddd | dppp | ssss | N, OV, Z, C |
| SL Ws,Wd | 1101 | 0000 | 0Bqq | qddd | dppp | ssss | N, Z, C |
| BTSTS.Z Ws,#bit4 | 1010 | 0100 | bbbb | Z000 | 0ppp | ssss | Z |
| BRA OV, Expr | 0011 | 0000 | nnnn | nnnn | nnnn | nnnn | none |
| BRA C, Expr | 0011 | 0001 | nnnn | nnnn | nnnn | nnnn | none |
| BRA N, Expr | 0011 | 0011 | nnnn | nnnn | nnnn | nnnn | none |
| BRA Z, Expr | 0011 | 0010 | nnnn | nnnn | nnnn | nnnn | none |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Encoding | OP | Flag N | Flag OV | Flag Z | Flag C | ALU OP | MEM Wr | MEM 2Reg | Reg Wr | Branch | Reg Dest | Wb  Reg |
| ADD | 01000 | 1 | 1 | 1 | 1 | 000 | 0 | 0 | 1 | 0 | 1 | 1 |
| SUB | 01010 | 1 | 1 | 1 | 1 | 001 | 0 | 0 | 1 | 0 | 1 | 1 |
| AND | 01100 | 1 | 0 | 1 | 0 | 011 | 0 | 0 | 1 | 0 | 1 | 1 |
| IOR | 01110 | 1 | 0 | 1 | 0 | 010 | 0 | 0 | 1 | 0 | 1 | 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 | 101 | - | - |
| LSR Wb,Wns,Wnd | 11011 | 1 | 0 | 1 | 0 | 100 | 0 | 0 | 1 | 0 | 1 | 0 |
| NEG Ws,Wd | 11101 | 1 | 1 | 1 | 1 | 101 | 0 | 0 | 1 | 0 | 1 | 1 |
| SL Ws,Wd | 11010 | 1 | 0 | 1 | 1 | 110 | 0 | 0 | 1 | 0 | 1 | 1 |
| BTSTS.Z Ws,#bit4 | 10100 | 0 | 0 | 1 | 0 | 111 | 0 | 0 | 1 | 0 | 1 | 1 |
| BRA OV,Expr | 00110 | 0 | 0 | 0 | 0 | - | 0 | 0 | 0 | 010 | - | - |
| BRA C,Expr | 00110 | 0 | 0 | 0 | 0 | - | 0 | 0 | 0 | 011 | - | - |
| BRA N,Expr | 00110 | 0 | 0 | 0 | 0 | - | 0 | 0 | 0 | 100 | - | - |
| BRA Z,Expr | 00110 | 0 | 0 | 0 | 0 | - | 0 | 0 | 0 | 001 | - | - |