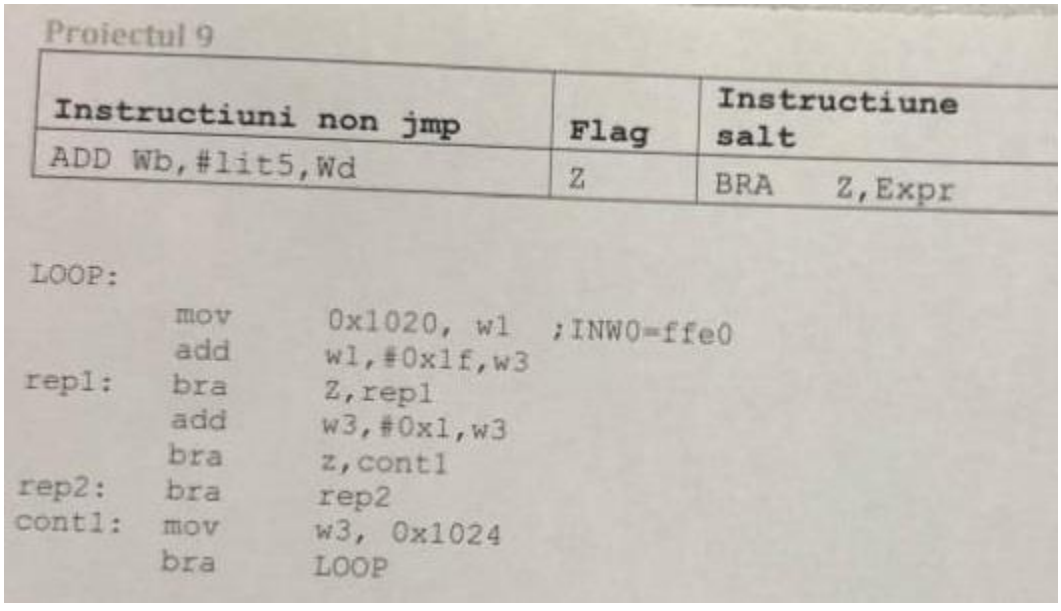
PROCESOR PIC24

DOCUMENTAȚIE

Proiectul 9

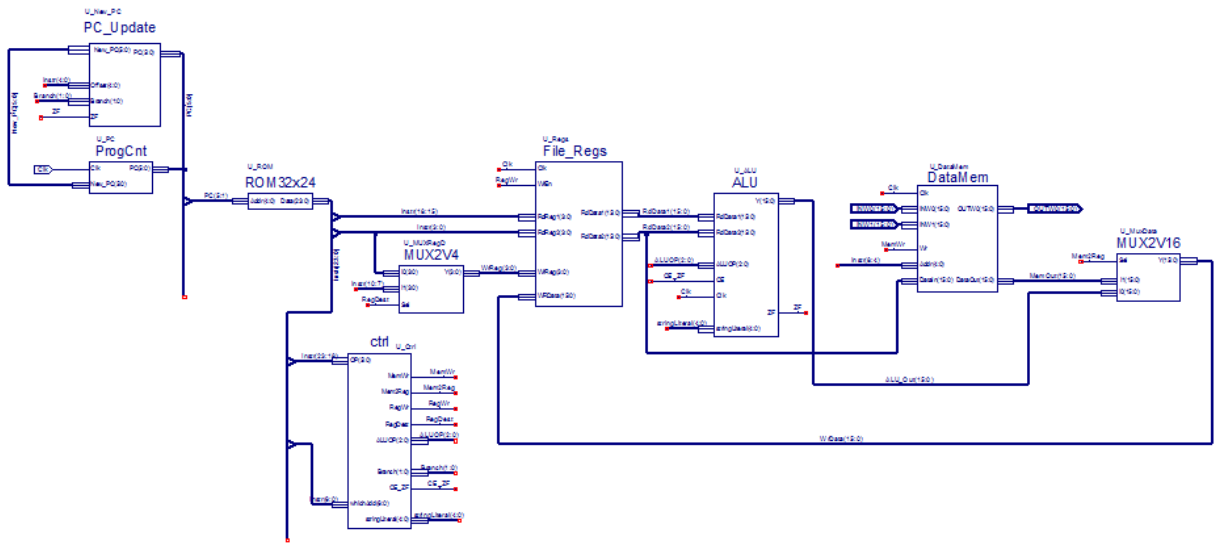


Proiectul 9

Instructiuni non jmp	Flag	Instructiune salt
ADD Wb,#lit5,Wd	Z	BRA Z,Expr

```
LOOP:
        mov     0x1020, w1   ;INW0=ffe0
        add     w1,#0x1f,w3
rep1:   bra     Z,rep1
        add     w3,#0x1,w3
        bra     z,cont1
rep2:   bra     rep2
cont1:  mov     w3, 0x1024
        bra     LOOP
```

Schema bloc a procesorului:



PC_Update – incrementeaza ProgCnt(Program Counter) - ul cu 2 sau cu 2 + offset + 2 daca daca intalneste o instructiune de tip Branch si conditiile impuse la actualizarea PC-ului sunt indeplinite.

- **Z**-flag-ul de zero
- **Branch**- semnalul pentru instructiunile de branch
- **Offset**- valoarea offset-ului (ultimul bit din pc este mereu 0 la PIC 24, de aceea avem nevoie de offset)
- **New_PC**- Program Counter-ul calculat
- **PC**- Program Counter-ul curent

ProgCnt - Program Counter-ul care ia valoarea incremenetata de PC_Update(New_PC) doar cand Clock-ul este ‘high’, adica are valoarea ‘1’;

- **Clk**- semnalul de Clock
- **New_PC**- noul Program Counter
- **PC**- Program Counter-ul curent

ROM32x24 - reprezinta memoria de program.

-are maxim 32 de instructiuni, iar fiecare este pe 24 de biti.

-in proiect exista 4 memorii: una pentru secventa de instructiuni A(all), una pentru flagul proiectului (Z), una pentru secventa de instructiuni de pe bilet(bilet), si ultima pentru testarea pe placa, care excuta o operatie simplu de urmarit si o singura afisare pe ecran.

-iesirea acestui bloc este codul pentru instructiunea selectata. Aceasta instructiune se selecteaza cu ajutorul numarului dat de ProgCnt

- **Addr**(4:0) - adresa de unde selectam instructiunea
- **Data**(23:0)- instructiunea (iesirea blocului)

MUX2V4 - din codul de instructiuni, bitii 3-0 sunt cei folositi pentru selectarea registrului destinatie pentru instructiunea MOV f,wnd;

-celelalte instructiuni care au registru destinatie, bitii folositi pentru selectarea registrului sunt bitii 10-7;

-acest MUX are ca iesire, in functie de semnalul RegDest, bitii folositi pentru selectarea registrului destinatie al instructiunii curente;

- **I1(3:0)** - bitii pentru instructiunea MOV f,wnd
- **I0(3:0)** - bitii pentru celelalte instructiuni
- **Sel** - bitul de selectie
- **Y(3:0)** - bitii folositi pentru selectarea registrului de destinatie

Ctrl – din blocul Ctrl provin valorile semnalelor folosite, valori luate in functie de OPCODE-ul instructiunii;

- **CE_ZF** - '1' pentru instructiunile care afecteaza flag-ul
  - folosit in ALU
- **MemWr** – semnal pentru determinarea daca o instructiune scrie in memorie sau nu

- **Mem2Reg** – daca este ‘1’, inseamna ca data va fi scrisa din memorie in registru (Mov f,wnd), altfel, data iesita din ALU va fi scrisa in registru
- **RegWr** – semnal care indica daca data se scrie in registru
- **RegDest** – semnal care indica daca registrul destinatie este pe bitii de instructiune 3-0 sau 7-10, daca acest semnal este 0, mux-ul 2v4 va alege intrarea I0, daca este 1, va alege I1.
- **ALUOP(2:0)** - determina operatia in ALU

File-Regs - registrele generale, contine 16 registre(W0-W15)

-fiecare registru are 16 biti

-are doua porturi de citire si un port de scriere, deoarece exista instructiuni care necesita simultan citirea a doua registre si scrierea unui registru

- **Clk** – semnalul de clock
- **WrEn** – semnal care activeaza scrierea
- **RdReg1(3:0)** – biti de date
- **RdReg2(3:0)** - biti de date
- **WrReg(3:0)** - semnal care selecteaza registrii
- **WrData(15:0)** - port de scriere
- **RdData1(15:0)** - port de citire
- **RdData2(15:0)** - port de citire

ALU – implementeaza instructiunile de adunare, scadere, and, or sau altele si calculeaza flag-ul Z (flag care ne spune daca rezultatul unei operatii este zero sau nu) .

- **RdData1(15:0)** - operand de intrare
- **RdData2(15:0)** - operand de intrare
- **ALUOP(1:0)** - operatia realizata de ALU
- **Clk** - semnalul de Clock
- **CE_ZF** - semnalul care indica flag-ul Z

- **Z** – flag-ul Z
- **Y** - operand de iesire

DataMem – blocul de memorie RAM

- contine un array de 16 elemente a cate 16 biti fiecare(aici se scrie dupa executia lui MOV wns,f

- **Clk** - semnalul de clock
- **INW0(15:0)** - adresa de citire
- **INW1(15:0)** - adresa de citire
- **Wr(MemWr)** - semnalul care indica scrierea in memorie
- **Addr(4:0)** - bitii care indica adresa
- **DataIn(15:0)** - valoarea care trebuie scrisa
- **DataOut(15:0)** - primeste INW0 sau INW1
- **OUTW0(15:0)** - adresa de scriere

MUX2V16 – decide cu ajutorul semnalului Mem2Reg daca datele scrise in registru provin din ALU sau din memorie in cazul instructiunii MOV f,wnd

- **Sel(Mem2Reg)** - selectia, daca este '1', se selecteaza data provenita din memorie
- **I0(15:0)** - data din ALU
- **I1(15:0)** - data din memorie
- **Y(15:0)** - iesirea

Tabela de Codificari:

Encoding	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0	Flags
ADD	0100	0www	wBqq	qddd	dppp	ssss	Z
SUB	0101	0www	wBqq	qddd	dppp	ssss	Z
AND	0110	0www	wBqq	qddd	dppp	ssss	Z
IOR	0111	0www	wBqq	qddd	dppp	ssss	Z
MOV f, wnd	1000	0fff	ffff	ffff	ffff	dddd	-
MOV wns, f	1000	1fff	ffff	ffff	ffff	ssss	-
BRA expr	0011	0111	nnnn	nnnn	nnnn	nnnn	
BRA Z, expr	0011	0010	nnnn	nnnn	nnnn	nnnn	-
ADD wb, #lit5,wd	0100	0www	wBqq	qddd	d11k	kkkk	Z

Tabela de Adevar:

	OPCODE	CE_ZF	ALUOP	MemWr	Mem2Reg	RegWr	RegDest
ADD	01000	1	000	x	x	1	1
SUB	01010	1	001	x	x	1	1
AND	01100	1	010	x	x	1	1
IOR	01110	1	011	x	x	1	1
MOV f,wnd	10000	0	x	x	1	1	0
MOV wns,f	10001	0	x	1	x	x	x
BRA Expr	001101	0	x	x	x	x	x
BRA Z,Expr	001100	0	x	x	x	x	x
ADD wb,#lit5,wd	01000	1	100	x	x	1	1