PIC 24 PROJECT

NUMBER 5

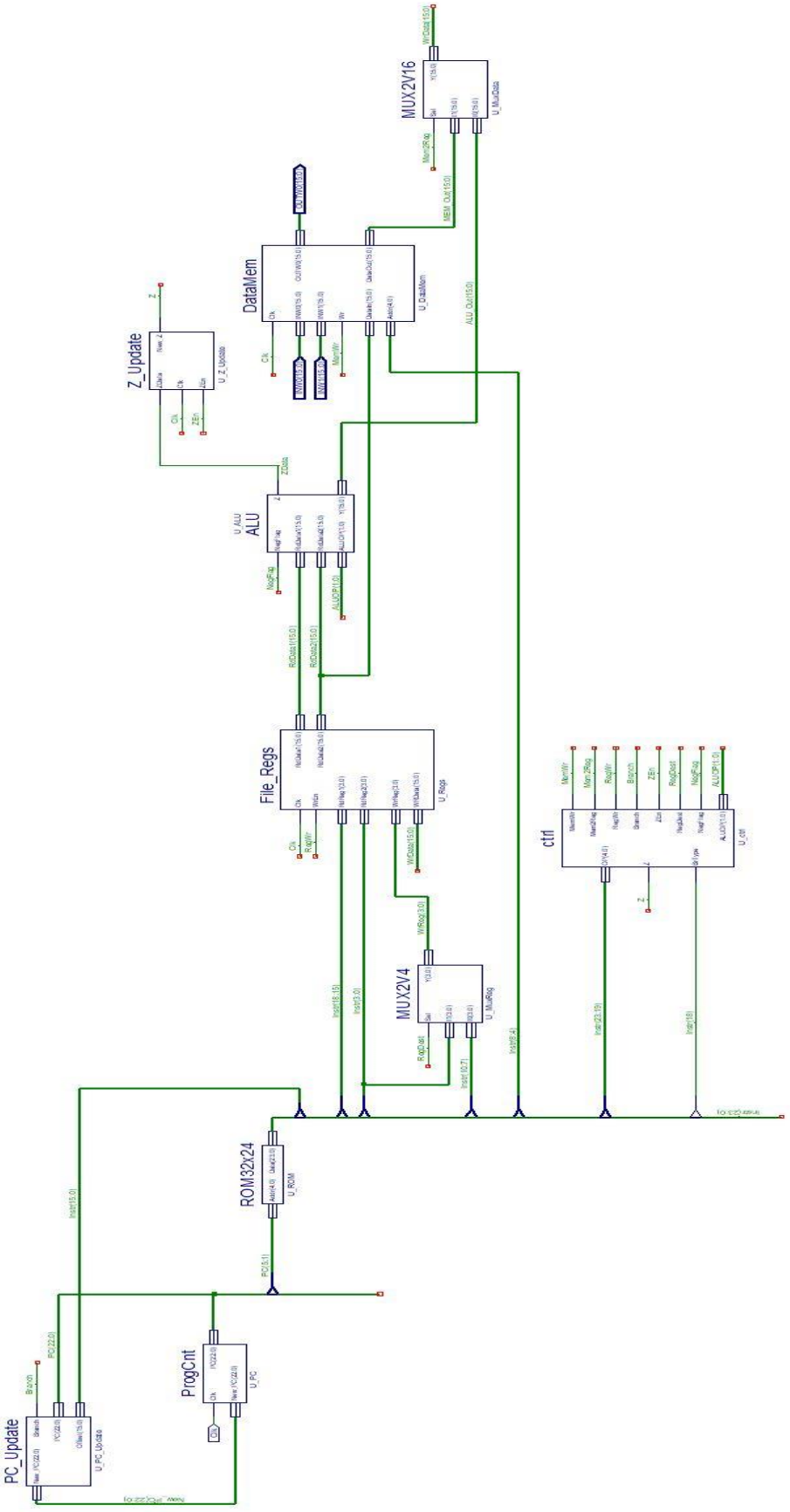
Buşe-Dragomir Alexandru

Dană Andrei Iulian

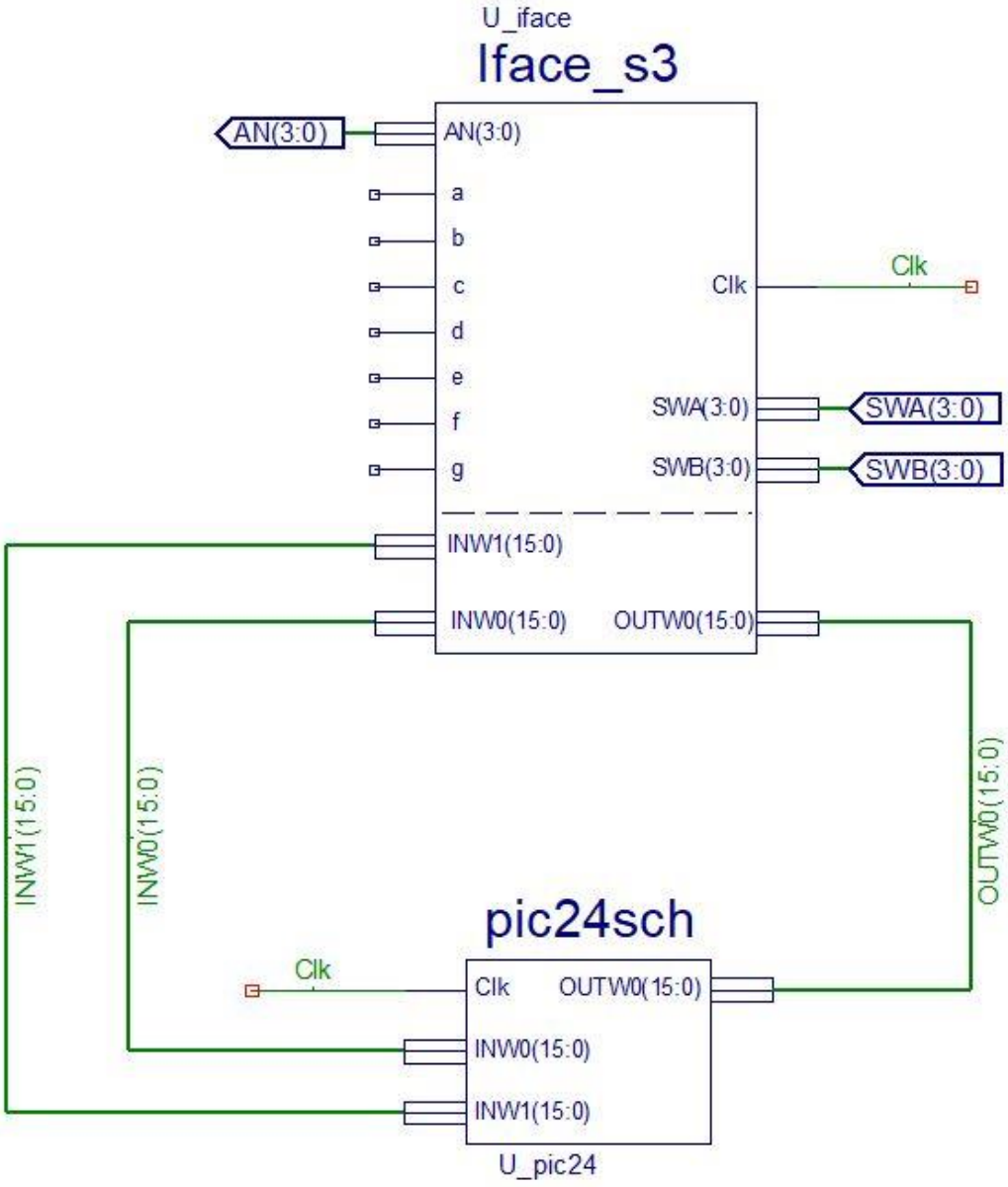
CEN 3.1 A

05.02.2019

PIC 24 PROJECT SCHEME



PIC 24 BOARD SCHEME

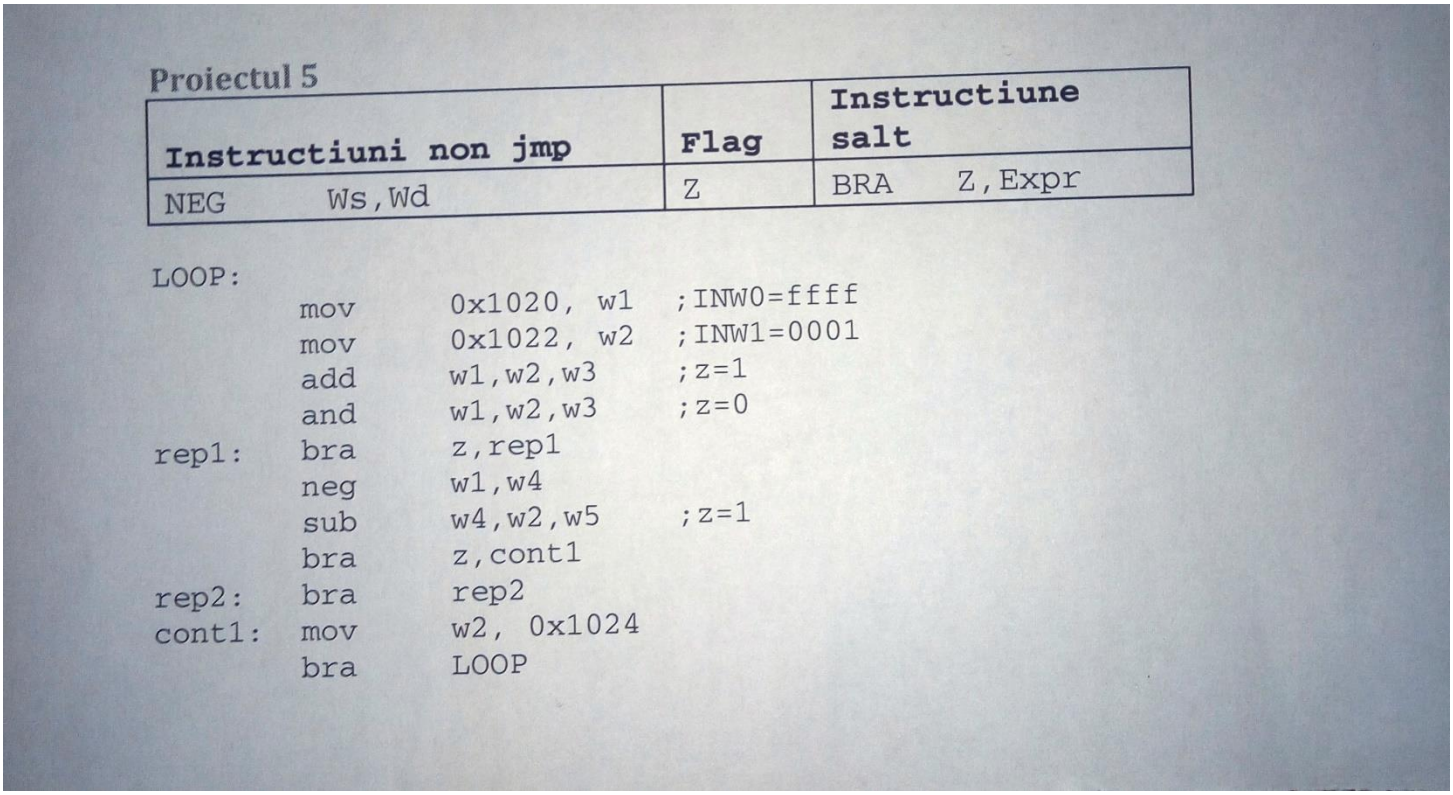


Project number 5 instructions:

Non-jump instructions: NEG Ws, Wd

Flag: Z

Jump instructions: BRA Z, Expr



Proiectul 5

Instructiuni non jmp	Flag	Instructiune salt
NEG Ws,Wd	Z	BRA Z,Expr

```
LOOP:
        mov     0x1020, w1   ;INW0=ffff
        mov     0x1022, w2   ;INW1=0001
        add     w1,w2,w3     ;z=1
        and     w1,w2,w3     ;z=0
rep1:   bra     z,rep1
        neg     w1,w4
        sub     w4,w2,w5     ;z=1
        bra     z,cont1
rep2:   bra     rep2
cont1:  mov     w2, 0x1024
        bra     LOOP
```

Encoding	23-20	19-16	15-12	11-8	7-4	3-0	Flags
ADD Wb,Ws,Wd	0100	0www	wBqq	qddd	dppp	ssss	N, OV, **Z**, C
SUB Wb,Ws,Wd	0101	0www	wBqq	qddd	dppp	ssss	N, OV, **Z**, C
AND Wb,Ws,Wd	0110	0www	wBqq	qddd	dppp	ssss	N, -, **Z**, -
IOR Wb,Ws,Wd	0111	0www	wBqq	qddd	dppp	ssss	N, -, **Z**, -
MOV F, WND	1000	0fff	ffff	ffff	ffff	dddd	-
MOV WNS, F	1000	1fff	ffff	ffff	ffff	ssss	-
BRA	0011	0111	nnnn	nnnn	nnnn	nnnn	-
BRA Z	0011	0010	nnnn	nnnn	nnnn	nnnn	-
NEG Ws, Wd	1110	1010	0Bqq	qddd	dppp	ssss	N, OV, **Z**, C

The ‘w’ bits select the address of the base register.

The ‘B’ bit selects byte or word operation (‘0’ for word, ‘1’ for byte).

The ‘q’ bits select the destination Address mode.

The ‘d’ bits select the destination register.

The ‘p’ bits select the source Address mode.

The ‘s’ bits select the source register.

The ‘n’ bits are a signed literal that specifies the number of program words offset from (PC + 2).

The ‘f’ bits select the address of the file register.

PIC24 Data Memory

Adresă Cuvânt	Adresa Octet Superior (biţii 15..8)	Adresa Octet Inferior (biţii 7..0)
0000h	0001h	0000h
0001h	0003h	0002h
0002h	0005h	0004h
...	...	...
7ffeh	fffdh	fffch
7fffh	ffffh	fffeh

PROGRAM COUNTER

The Program Counter (PC) is 23 bits wide. Instructions are addressed in the 4M x 24-bit user program memory space by PC<22:1>, where PC<0> is always set to '0' to maintain instruction word alignment and provide compatibility with data space addressing. This means that during normal instruction execution, the PC increments by 2.

Program memory located at 0x800000 and above is utilized for device configuration data, Unit ID and Device ID. This region is not available for user code execution and the PC can not access this area. However, one may access this region of memory using table instructions. For details on accessing the configuration data, Unit ID, and Device ID, refer to the specific device family reference manual.

Blocul de memorie pentru PIC24 este foarte asemănător cu blocul de memorie de la MIPS.

Blocul memoriei de date conţine o memorie RAM alcătuită din 16 locaţii de 16 de biţi plus 3 locaţii speciale. Acest bloc foloseşte 5 adrese. Adresele A4-A0 generate de procesor (dreptunghiul albastru din figura de mai sus) se conectează pinii Addr(4:0) ai blocului de memorie. Celelalte adrese procesor nu contează. Din acest motiv blocul memoriei de date ocupă multiple spaţii de adresă procesor.

În asamblare se vor folosi adresele procesor specificate mai jos:

- Cele 16 locaţii de 16 biţi ocupă spaţiul de adrese de octet **1000h -101Fh**
- Cuvântul de la adresa de octet **1020h** este de tipul **„Read only”**. **Întotdeauna** data citită din această locaţie are valoarea marcherului I/O de intrare **INW0**. Această locaţie este folosită pentru introducerea de date. Scrierea acestei locaţii nu are efect: data scrisă se va pierde.
- Locaţia de la adresa **1022h** este de tipul **„Read only”**. **Întotdeauna** data citită din această locaţie are valoarea marcherului I/O de intrare **INW1**. Această locaţie este folosită pentru introducerea de date. Scrierea acestei locaţii nu are efect: data scrisă se va pierde.
- Locaţia de la adresa **1024h** este de tipul **„Write only”**. Data scrisă în această locaţie va apare în exteriorul sistemului prin intermediul marcherului I/O de ieşire **OUTW0**. Această locaţie este folosită pentru extragerea de date. **Întotdeauna** data citită din această locaţie este nedefinită. Data citită din această locaţie **NU** coincide cu data scrisă anterior.

	Opcode	ALUOP	MEM2REG	MEMWR	REG WR	REGDEST	BRANCH	ZEn	NegFlag
ADD Wb,Ws,Wd	01000	00	0	0	1	0	0	1	0
SUB Wb,Ws,Wd	01010	01	0	0	1	0	0	1	0
AND Wb,Ws,Wd	01100	10	0	0	1	0	0	1	0
IOR Wb,Ws,Wd	01110	11	0	0	1	0	0	1	0
MOV F, WND	10000	00	1	0	1	1	0	0	0
MOV WNS, F	10001	00	0	1	0	0	0	0	0
BRA	00110	00	0	0	0	0	1	0	0
BRA Z	00110	00	0	0	0	0	1	0	0
NEG Ws, Wd	11101	00	0	0	1	0	0	1	1

Tabelul de adevăr pentru semnalele ce apar la nivelul blocului de control (ctrl)

ZEn este semnalul prezentat ca CE în indicațiile din materialele auxiliare; când are valoarea ‘0’ logic, indicatorul Z nu își va schimba valoarea. În cazul în care ZEn este ‘1’, Z va putea să își schimbe starea.

Descrierea semnalelor utilizate

- ALUOP: semnal pe 2 biți ce reprezintă operația pe care ALU o va efectua
- Branch: este setat pe 1 dacă urmeaza o intrucțiune Branch, Expr sau dacă urmează o instrucțiune Branch Z, Expr și Z=1
- ZEn: este setat pe 1 când operația curentă poate activa flagul Z
- WrReg: conține poziția registrului în care valoarea trebuie scrisă
- Mem2Reg: selectează ce urmează să fie scris in registru (dintre operația din ALU si o valoare citită din memoria de date)
- ZData: ține valoarea lui Z pentru operația curentă
- PC: numărător de program (program counter), pointează către următoarea instrucțiune ce va fi executată (din memoria ROM)
- SWA/SWB: cei mai nesemnificativi 4 biți din INW0 și INW1
- BrType: valoarea 1 când urmeaza BRA, Expr și 0 când urmează BRA Z, Expr
- RdData1/RdData2: conținutul celor doi regiștri citiți, care vor reprezenta cei doi operanzi din ALU
- Clk: impuls de tact pentru sincronizarea blocurilor
- Z: flagul zero

- New_Z: noua valoare a lui Z la ieșirea din blocul Z_Update
- Instr: contine forma binara a instrucțiunii din ROM
- RegDest: spune daca se va scrie in registrul sursă (Ws) sau in registrul destinație (Wd)
- MemWr: spune dacă rezultatul operației se va stoca în memorie
- RegWr: spune dacă rezultatul operației se va stoca într-unul dintre registre (cel pointat de către WrReg)
- INWO: I/O marker pentru furnizarea de date (este read only la nivelul memoriei RAM)
- INW1: I/O marker pentru furnizarea de date (este read only la nivelul memoriei RAM)
- OUTW0: I/O marker pentru observarea în exterior a conținutului din memorie (la nivelul RAM, este write only)
- NegFlag: flag setat de ctrl și primit de ALU care îi permite acestuia să determine dacă să execute o operație aritmetică sau să returneze complementul operandului al doilea (Ws), care va fi ulterior salvat în Wd (operatia NEG Ws, Wd)

Rolul blocurilor:

1. ALU: este responsabil cu realizarea operațiilor aritmetice si logice (ce se stabilesc decodificând ALUOP) asupra RdData1 si RdData2. ALU transmite rezultatul prin Y. De asemenea acesta trebuie să seteze flagul Z la 1 logic atunci când rezultatul operației realizate de ALU este zero. De asemenea, ALU va fi utilizat și pentru operația NEG Ws, Wd, generând complementul lui Ws (operandul 2 al ALU)

2. ProgCnt: va stoca valoarea din PC_Update pe frontul crescător al impulsului de tact

3. PC_Update: setează o valoare nouă pentru ProgCnt, fie prin incrementarea cu 2, fie prin adăugarea offsetului înmulțit cu 2 (shiftat la stânga cu o poziție) pentru instrucțiunile de branch

4. CTRL: dându-se o instrucțiune , acesta va determina valorile tuturor semnalelor de control ce vor fi folosite de celelalte blocuri

5. ROM32x24: reprezintă memoria ROM si conține instrucțiunile programului; este organizată sub forma a 32 de vectori a câte 24 de biți fiecare

6. MUX2V4: are rolul de a selecta adresa registrului destinație în funcție de specificațiile din instrucțiune (poate fi Ws sau Wd). Wd pentru instrucțiunile aritmetice și NEG, iar Ws pentru MOV F, WND.
7. DataMem: o memorie RAM alcătuita din 16 locații, fiecare având 16 biți, doua locații read-only (INW0 si INW1) , o locație write-only (OUTW0)

8. File_Regs: conține toate registrele și se ocupă cu stocarea informației la nivelul acestora, cât și cu citirea din ele

9. MUX2V16: determină dacă data ce trebuie scrisă în registru provine de la blocul de memorie sau de la ALU (în funcție de instrucțiune)

10. Z_Update: se ocupă cu updatarea valorii lui Z dacă semnalul ZEn este activ și numai pe frontul crescător al impulsului de tact