4.2 Unitatea aritmetica - logica

UAL executa toate operatiile aritmetice si logice din calculator, iar rezultatele sunt depuse in memorie sau trimise la unitatea de iesire pentru a fi furnizate in exterior. O unitate aritmetica – logica poate fi reprezentata prin simbolul (fig.4.2.1):

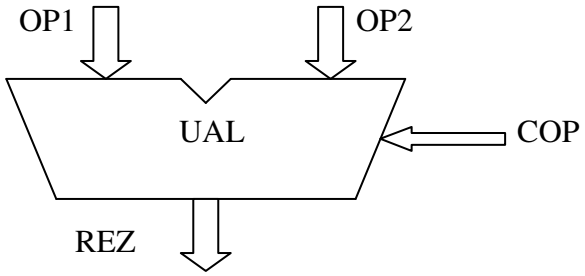

Fig.4.2.1 Unitate aritmetica – logica.

unde,

OP1, OP2 sunt cei doi operanzi reprezentati fiecare pe *n* biti;

REZ este rezultatul operatiei, de asemenea reprezentat pe *n* biti;

COP este codul de selectie a operatiei (codul operatiei), reprezentat pe *m* biti, deci se pot codifica in total 2^m operatii diferite.

UAL poate fi conectat intr-un sistem de calcul cu celelalte unitati functionale in diferite moduri:

Magistrala unica. Sistemul utilizeaza o magistrala unica, accesul la UAL realizandu-se prin intermediul a doua registre temporare, T1 si T2, inaccesibile programatorului (fig.4.2.2). Cei doi operanzi se incarca succesiv de pe magistrala in registrele temporare, iar rezultatul produs de UAL este plasat pe magistrala pentru a fi transferat la destinatie intr-un registru accesibil prin program, locatie de memorie sau port de iesire.

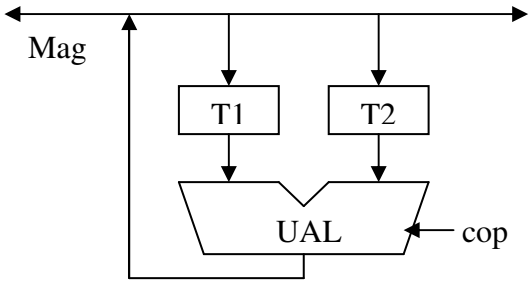

Fig.4.2.2 Conectarea UAL prin registre temporare.

Este cea mai simpla solutie, dar si cea mai putin eficienta. Astfel, in cazul cel mai defavorabil, cand cei doi operanzi si rezultatul sunt locatii de memorie, o operatie completa la nivelul UAL necesita trei cicluri succesive de memorie, doua citiri si o scriere.

Registru accumulator. Solutia utilizeaza un registru accumulator (Acc) in care se incarca de pe magistrala unul dintre operanzi (fig.4.2.3). Al doilea operand este furnizat de la sursa (registru utilizator, locatie de memorie sau port de I/E) direct prin intermediul magistralei. Rezultatul obtinut este incarcat in accumulator si poate reprezenta un operand pentru operatia urmatoare. Rezultatul final se va transfera din accumulator la destinatie, prin intermediul magistralei.

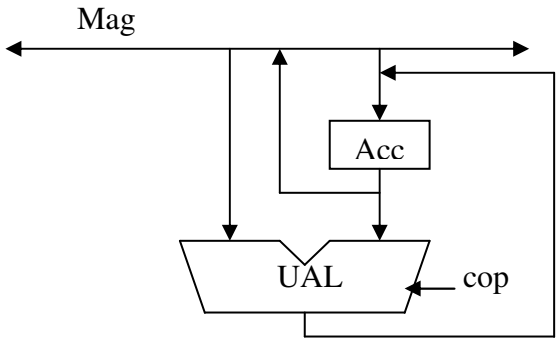

Fig.4.2.3 Conectarea UAL prin registru accumulator.

Aceasta solutie ofera o crestere a vitezei de executie a instructiunilor aritmetice – logice.

Interconectare cu trei magistrale. Aceasta solutie (fig.4.2.4) utilizeaza trei magistrale pentru furnizarea operanzilor (Mag A si Mag B) si pentru preluarea rezultatului (Mag C).

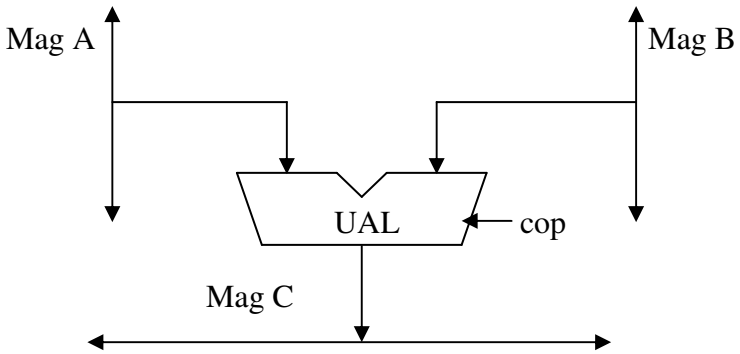

Fig.4.2.4 Conectarea UAL prin trei magistrale.

Chiar daca solutia are un cost ridicat, ofera in schimb viteza mare in prelucrarea operanzilor. Astfel, in cazul cel mai defavorabil, cand cei doi operanzi si rezultatul sunt locatii de memorie, dar in module independente conectate fiecare la cate o magistrala, o operatie completa la nivelul UAL necesita, in medie, un singur ciclu de memorie.

In continuare se vor studia cateva exemple de unitati aritmetice – logice.

Sumator-scazator pentru numere in cod complementar

Aceasta unitate aritmetica permite executia operatiilor de adunare si scadere pentru operanzi reprezentati in virgula fixa, cod complementar. Structura acestei unitatii este reprezentata in urmatoarea schema bloc (fig.4.2.5):

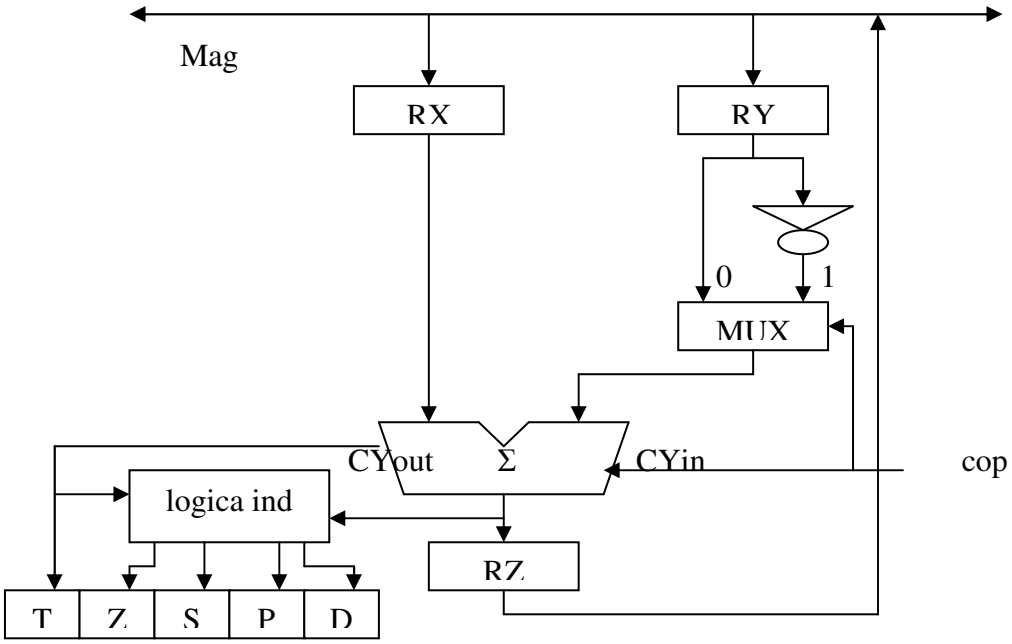

Fig.4.2.5 Sumator / scazator pentru cod complementar.

Resursele unitatii sumator – scazator sunt:

RX , RY sunt doua registre temporare (de tipul T1 si T2 din solutia generala precedenta), in care se incarca succesiv de pe magistrala cei doi operanzi, X si Y, pe *n* biti fiecare.

RZ este un registru care preia rezultatul si il plaseaza apoi pe magistrala pentru a fi transferat la destinatie.

Σ este un sumator, efectuand intotdeauna operatia de insumare.

MUX este un bloc de *n* multiplexoare 2:1 comandat de codul de selectie a operatiei *cop*, un singur bit (*cop* = 0 adunare, *cop* = 1 scadere). Pe intrarile 0 ale multiplexorului se conecteaza al doilea operand in forma directa (Y), iar pe intrarile 1, al doilea operand in forma complementata (Y#). Acelasi semnal de selectie *cop* este conectat si pe intrarea de transport a sumatorului *CYin*, care se aduna in rangul c.m.p.s. la o operatie efectuata. Astfel,

cop = 0 => Z = X + Y + 0 = X + Y (adunare)

cop = 1 => Z = X + Y# + 1 = X –Y (scadere).

Logica ind este o logica combinationala care pozitioneaza cativa bistabili, numiti indicatorii de conditie. Acesti indicatori se pozitioneaza in 1 sau 0 dupa fiecare operatie, reflectand caracteristici ale rezultatului. Astfel:

T este indicatorul de transport si se pozitioneaza in *T* = 1 daca exista transport de la rangul c.m.s al rezultatului si in *T* = 0 daca nu exista transport (se observa in schema ca acest bistabil este pozitionat direct de catre iesirea de transport a sumatorului, *CYout*);

Z este indicatorul pentru rezultat zero si se pozitioneaza in *Z* = 1 pentru rezultat nul si *Z* = 0 pentru rezultat diferit de zero;

S este indicatorul de semn, fiind pozitionat de catre bitul c.m.s. al rezultatului, care este bitul de semn. Astfel, *S* = 1 pentru rezultat negativ si *S* = 0 pentru rezultat pozitiv;

P este indicatorul de paritate. Se poate utiliza paritatea para sau impara. In cazul utilizarii paritatii impare, acest indicator se pozitioneaza *P* = 1 daca suma modulo 2 peste toti bitii rezultatului este 0 si *P* = 0 in caz contrar;

D este indicatorul de depasire. Asa cum s-a studiat in capitolul 2, un transport la adunare in cazul numerelor reprezentate in cod complementar nu inseamna rezultat eronat, deci era necesar un alt indicator care sa semnaleze cazurile de depasire. Acest indicator se pozitioneaza *D* = 1 daca exista depasire (rezultat incorect) si *D* = 0 daca nu exista depasire (rezultat corect).

Unitate de inmultire in cod direct

Aceasta unitate aritmetica executa operatia de inmultire a doua numere reprezentate in virgula fixa cod direct. Schema bloc a dispozitivului este prezentata in figura urmatoare (fig.4.2.6):

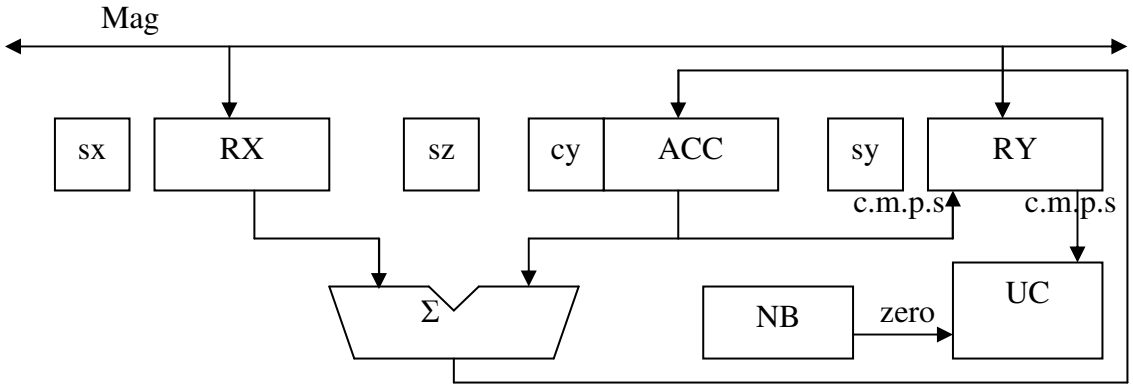

Fig.4.2.6 Unitate de inmultire pentru cod direct.

Unitatea de inmultire contine urmatoarele resurse:

RX registru pentru modulul deinmultitului;

sx bistabil pentru semnul deinmultitului;

RY registru pentru modulul inmultitorului;

sy bistabil pentru semnul inmultitorului;

sz bistabil pentru semnul rezultatului;

Σ sumator;

ACC registru accumulator pentru pastrarea rezultatelor partiale, iar in final va contine bitii c.m.s. ai modulului produsului;

cy bistabil pentru memorarea transportului furnizat de sumator;

NB numarator de biti pentru numararea pasilor operatiei de inmultire (inmultirea se executa intr-un numar de pasi egal cu numarul de biti folositi la reprezentarea modulului fiecarui operand);

UC unitate de comanda.

La inceputul operatie se incarca de pe magistrala, succesiv, cei doi operanzi in sx,RX si sy,RY, iar in ACC se incarca 0. Registrele cy,ACC si RY sunt registre cu deplasare, informatia poate fi deplasata o pozitie spre dreapta, cu trecerea bitului c.m.p.s. din ACC in RY (registrele sunt concatenate). La fiecare pas, unitatea de comanda testeaza bitul curent al inmultitorului, care este adus prin deplasari in pozitia c.m.p.s. din registrul RY. Daca acest bit este 1 se aduna prin intermediul sumatorului deinmultitul RX cu rezultatul partial ACC, rezultatul fiind memorat in ACC, iar apoi se executa o deplasare cu o pozitie spre dreapta pentru cy,ACC,RY. Daca bitul curent al inmultitorului este 0, nu se mai face operatia de adunare, dar se executa deplasarea. Numaratorul de biti este initializat la inceputul operatiei cu valoarea n-1 (numarul de biti pentru reprezentarea modulului fiecarui operand) si este decrementat la fiecare

pas. Cand numaratorul ajunge in zero, anunta unitatea de comanda prin activarea unui semnal (*zero*) incheierea operatiei de inmultire. Rezultatul este obtinut pe lungime dubla in ACC,RY. In continuare, este prezentata o descriere precisa in pseudocod pentru functionarea unitatii de inmultire.

```
sx,RX ← X  //deinmultitul pe n biti
sy,RY ← Y // inmultitorul pe n biti
cyy,ACC ← 0
NB ← n–1   //numar de pasi
sz ← sx ⊕ sy    //semnul rezultatului
repeta
|       daca RY0 = 1 atunci
|       └      cyy,ACC ← RX + ACC
|       cyy,ACC,RY ← cyy,ACC,RY*2^-1   //deplasare dreapta o pozitie
└       NB ← NB – 1
cat timp NB ≠ 0
Z = sz,ACC,RY   //rezultatul pe 2n-1 biti, lungime dubla
```

Unitate logica

In principiu o unitate logica genereaza rezultatele tuturor operatiilor logice posibile, iar rezultatul final este selectat prin multiplexare pe baza codului de operatie. In acest paragraf este descrisa o unitate logica foarte simpla care poate executa patru operatii logice cu operanzi pe 4 biti fiecare. Deci sunt necesari doi biti pentru reprezentarea codului operatiei *cop*. Aceste operatii sunt prezentate in tabela urmatoare:

cop	Operatia
00	X SAU Y
01	X SI Y
10	NU X
11	X • 2

Ultima linie din tabela semnifica deplasare stanga o pozitie. Schema de principiu a acestei unitati logice este prezentata in figura urmatoare (fig.4.2.7):

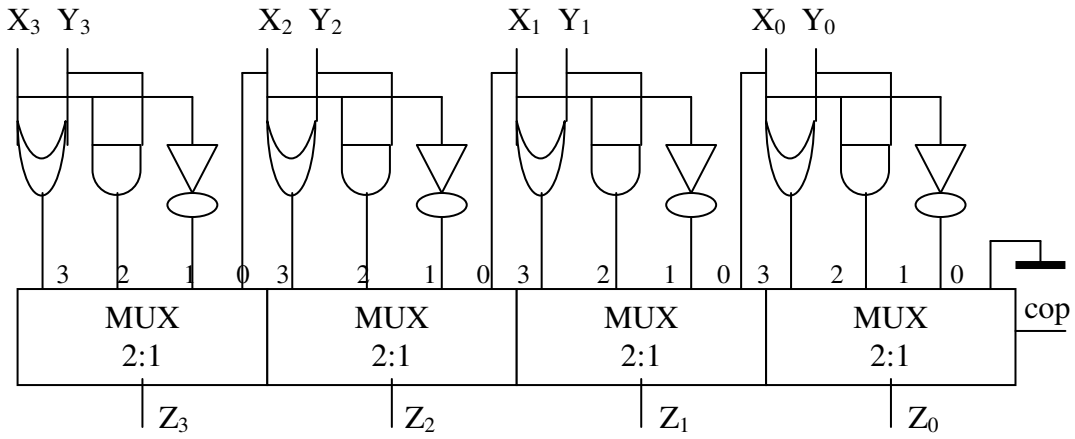

Fig.4.2.7 Unitate logica pe 4 biti.

Evident ca pentru operatiile logice SAU, SI, NU au fost utilizate porti logice de tipurile corespunzatoare. Operatia de deplasare s-a realizat prin decalarea conexiunilor, astfel in pozitia 3 s-a conectat bitul X_2, in pozitia 2 bitul X_1, ..., iar in pozitia 0 s-a conectat 0 logic (masa electrica), caci prin deplasare spre stanga se introduce un bit 0 prin dreapta numarului.