2. Microprocesorul 8086

2.1. Scurt istoric

Microprocesoarele "pe n biţi" sunt acele procesoare la care lungimea cuvântului prelucrat (dimensiunea regiştrilor, lungimea uzuală a unui operand) este de n biţi. Cele mai răspândite microprocesoare în calculatoarele personale sunt cele din familia INTEL. În continuare este prezentată o evoluţie a acestei familii:

8080/8085	microprocesor pe 8 biţi, 64Ko de memorie
8086/8088	microprocesor pe 16 biţi maximum 1 Mo de memorie, mod de lucru real, single task arhitectură de prelucrare secvenţial-paralelă (pipeline)
80286	magistrală internă şi externă pe 16 biţi maximum 16 Mo de memorie, mod de lucru real/protejat suport pentru sistemele de operare multitasking
80386DX	microprocesor pe 32 de biţi, 4 Go de memorie mod de lucru real, protejat 16/32 de biţi, real virtual 8086 4GB de memorie principală posibil
80386SX	microprocesor pe 32 de biţi magistrală externă pe 16 biţi, 16 Mo de memorie
80486	dispune de coprocesor matematic FPU (Floating Point Unit) şi memorie cache pe acelaşi chip cu procesorul
Pentium	microprocesor pe 32 de biţi, 4 Go de memorie arhitectura superscalară (RISC) permite execuţia a mai mult de o instrucţiune/tact în anumite condiţii.

Caracteristicile diferiţilor reprezentanţi sunt prezentate în anexa 1.

2.2 Arhitectura software a microprocesorului 8086

Schema bloc internă a microprocesorului 8086 este prezentată în figura 4.3. Se compune din:

- EU – *Execution Unit* – Unitate de execuţie – execută calculele prin intermediul componentei ALU (UAL - Unitatea Aritmetică şi Logică); EU are în componenţă Unitatea Aritmetică şi Logică, regiştrii generali, regiştrii de adresare, regiştrii temporari, unitatea de decodificare şi comandă şi registrul de flaguri;
- BIU – *Bus Interface Unit* – Unitate de interfaţă cu bus-urile– componenta care pregăteşte execuţia fiecărei instrucţiuni; în esenţă, aceasta extrage o instrucţiune din memorie, o depune în coada de instrucţiuni şi calculează adresa din memorie a unui eventual operand.

Cele două componente lucrează în paralel în sensul că în timp ce EU execută instrucţiunea curentă, BIU pregăteşte instrucţiunea următoare; aceasta este o arhitectură secvenţial-paralelă.

Regiştrii generali sunt folosiţi pentru a stoca temporar operanzi şi rezultate în UC. Regiştrii de uz general AX, BX, CX, DX sunt regiştri pe 16 biţi. O utilizare implicită a acestor regiştri este următoarea:

- AX – registru acumulator
- BX – registru de bază în adresare
- CX – registru contor
- DX – registru de date (extensia acumulatorului), adresare indirectă a porturilor

Regiştrii pot fi accesaţi pe 16 biţi ca AX, BX, CX, DX sau pe 8 biţi, având partea inferioară AL, BL, CL, DL şi partea superioară AH, BH, CH, DH.
Regiştrii SP şi BP sunt regiştri destinaţi lucrului cu stiva. Stiva se defineşte ca o zonă de memorie (LIFO – Last In First Out) în care pot fi depuse valori, extragerea lor ulterioară facându-se în ordine inversă depunerii. SP – Stack Pointer – pointează spre ultimul element introdus în stivă, iar BP – Base Pointer – către baza stivei.
Regiştrii DI şi SI sunt regiştri index (Destination Index şi Source Index) destinaţi lucrului cu şiruri de octeţi sau cuvinte.
Registrul PSW/Flags este registrul indicatorilor de stare şi control. Un *flag* este un indicator reprezentat pe un bit. Indicatorii de stare ai registrului de flaguri sintetizează execuţia ultimei instrucţiuni. Pentru 8086 acest registru are 16 biţi din care sunt folosiţi numai 9. Structura acestui registru este prezentată mai jos.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	**O**	**D**	**I**	**T**	**S**	**Z**	-	**A**	-	**P**	-	**C**

Indicatorii de stare sunt:
C – Carry – indică un transport în afara domeniului de reprezentare a rezultatului;
P – Parity – este stabilit astfel încât împreună cu numărul de biţi de 1 din rezultat să rezulte un număr impar de cifre de 1;
A – Auxiliary – indică valoarea transportului de la bitul 3 la bitul 4 (între cifrele hexazecimale);
Z – Zero – are valoarea 1 dacă rezultatul ultimei instrucţiuni este egal cu zero;
S – Sign – are valoarea 1 când rezultatul ultimei operaţii este un număr strict negativ, adică copiază bitul MSB al rezultatului;
O – Overflow – indică depăşire de gamă: dacă rezultatul ultimei instrucţiuni a depăşit spaţiul rezervat rezultatului, în cazul operanzilor consideraţi numere cu semn.

$$O=C \oplus transport_{MSB-1\ MSB}$$

Indicatorii de control sunt:
T – Trap – flag pentru depanare; dacă are valoarea 1, atunci procesorul se opreşte după fiecare instrucţiune;
I – Interrupt – flag de întrerupere; permite sau invalidează acceptarea întreruperilor externe mascabile care apar pe intrarea INT a procesorului;
D – Direction – flag folosit la lucrul cu şiruri pentru a indica direcţia de parcurgere de-a lungul şirului, D=0 – adrese crescătoare, D=1 – adrese descrescătoare.

Exemple:

Valorile biţilor de Carry, Zero, Sign şi Overflow imediat după executarea următoarelor adunări de către un microprocesor pe 8 biţi sunt următoarele:

	Rezultat	**C**	**Z**	**S**	**O**
02 + 02	04	0	0	0	0
02 + 7D	7F	0	0	0	0
04 + 7F	83	0	0	1	1
80 + 80	00	1	1	0	1

2.3. Organizarea şi adresarea memoriei

Prin definiţie, *adresa unei locaţii de memorie* este numărul de ordine între începutul memoriei RAM şi locaţia respectivă. Dată fiind capacitatea de 1Mo a memoriei la 8086, o adresă trebuie să se reprezinte pe 20 de biţi, dar capacitatea regiştrilor şi a cuvintelor este de 16 biţi. Pentru rezolvarea situaţiei a apărut conceptul de segment de memorie, respectiv adresarea segmentată.

Segmentul de memorie reprezintă o succesiune continuă de octeţi care are următoarele proprietăţi: începe la o adresă multiplu de 16 (paragraf), are lungimea multiplu de 16 octeţi şi maximum 64 Kocteţi. Deoarece adresa de început a fiecărui segment este multiplu de 16, cei mai puţin semnificativi 4 biţi ai acestei adrese sunt zero.

Offset-ul sau deplasamentul reprezintă adresa unei locaţii faţă de începutul segmentului. Deoarece un segment are maximum 64 Ko, pentru exprimarea offsetului sunt suficienţi 16 biţi.

Adresa logică este o pereche de numere pe câte 16 biţi fiecare, unul reprezentând adresa de început a segmentului şi celălalt reprezentând offsetul în cadrul segmentului.

Adr.Logică = Reg.Segment:offset

Adresa fizică pe 20 de biţi se obţine din configuraţia de 16 biţi care localizează începutul segmentului înmulţită cu 16 la care se adună valoarea offsetului. Acest calcul este efectuat de unitatea de adresare din BIU.

$$AF_{20}=16 \bullet Reg.Segment_{16}+offset_{16}$$

Arhitectura 8086 permite existenţa a patru tipuri de segmente:

- CS – segment de cod – care conţine instrucţiuni;
- DS – segment de date – care conţine date care se prelucrează conform instrucţiunilor;
- SS – segment de stivă;
- ES – segment de date suplimentar (extrasegment).

În fiecare moment al execuţiei este declarat activ câte un singur segment din fiecare tip. Regiştrii CS (Code Segment), DS (Data Segment), SS (Stack Segment) şi ES (Extra Segment) din BIU reţin adresele de început ale segmentelor active, corespunzătoare fiecărui tip de segment.

Registrul IP – *Instruction Pointer* – conţine offsetul instrucţiunii curente în cadrul segmentului de cod curent, el fiind manipulat exclusiv de către BIU.

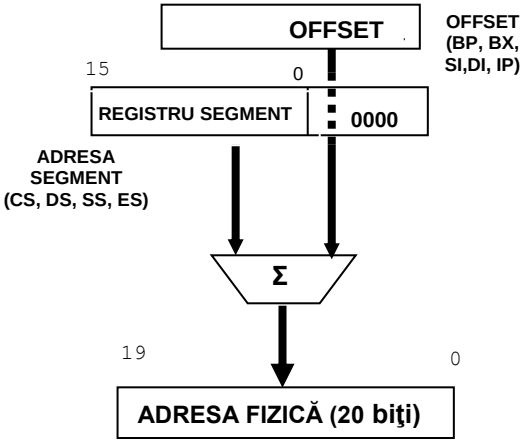


Fig.2.1. Calculul adresei fizice

Exemplu:

Dacă CS=24F6h şi IP=634Ah

- adresa logică este 24F6h:634Ah
- offsetul este 634Ah
- adresa fizică este 2B2AAh = 24F60h+634Ah

O adresă fizică poate fi obţinută din mai multe combinaţii de adrese logice:

Adresa logică (hexa)	Adresa fizică (hexa)
1000:5020	15020
1500:0020	15020
1502:0000	15020
1400:1020	15020
1302:2000	15020

O prezentare sintetică a regiştrilor microprocesorului 8086 se găseşte în tabelul următor.

Registru	Biţi	Nume registru
General	16	**AX, BX, CX, DX**
(date)	8	**AL, AH, BL,BH, CL, CH, DL, DH**
Pointer	16	**SP, BP**
Index	16	**SI, DI**
Segment	16	**CS, DS, SS, ES**
Instrucţiune	16	**IP** (contor instrucţiuni)
Flag-uri	16	**PSW**

Tabelul 2.1. Regiştrii procesorului 8086

2.4 Moduri de adresare

În cadrul unei instrucţiuni există mai multe moduri de a calcula adresa efectivă sau offsetul unui operand pe care aceasta îl solicită:

- modul registru – dacă operandul este un registru;
- modul imediat – atunci când în instrucţiune se află chiar valoarea operandului;
- modul de adresare cu memoria – dacă operandul se află undeva în memorie.

Instrucţiunile care au doi sau mai mulţi operanzi operează întotdeauna de la dreapta spre stânga. Operandul din dreapta este operandul sursă, el specifică datele care vor fi folosite, dar nu şi modificate. Operandul din stânga este operandul destinaţie, specifică datele care vor fi folosite şi modificate de către o anumită instrucţiune. Datele imediate nu sunt admise ca operand destinaţie.

Adresarea directă în cazul regiştrilor înseamnă folosirea valorii reale din interiorul registrului în momentul execuţiei instrucţiunii. În plus, există unele instrucţiuni care pot fi folosite numai cu operanzi regiştri şi instrucţiuni care pot fi folosite numai cu anumiţi regiştri. În cazul adresării cu regiştri memoria nu este accesată.

Exemple:

```
mov BX, DX      ; BX = DX
mov ES, AX      ; ES = AX
add AL, BH      ; AL = AL+BH
```

Observaţie: sursa şi destinaţia trebuie să aibă aceeaşi dimensiune !

Adresarea imediată are ca operand sursă o constantă. Acest mod nu poate fi folosit pentru încărcarea datelor în regiştri segment şi în flaguri.

Exemple:
```
mov AX, 2550h  ; AX = 2550h
mov CX, 625    ; CX = 625
add BL, 40h    ; BL = BL+40h
```

Când un operand se află în memorie, procesorul calculează adresa efectivă (offsetul) a datelor care vor fi prelucrate. Calcularea acestei adrese depinde de modalitatea în care este specificat operandul. Nu sunt admise operaţiile pentru care atât sursa cât şi destinaţia sunt operanzi din memorie.

A. Operandul cu adresare directă este o constantă sau un simbol care reprezintă adresa (segment şi deplasament) unei instrucţiuni sau a unor date. Deplasamentul unui operand cu adresare directă este calculat în momentul asamblării. Adresa fiecărui operand raportată la structura programului este calculată în momentul editării de legături. Adresa efectivă este calculată în momentul încărcării programului pentru execuţie. Adresa efectivă este întotdeauna raportată la un registru de segment.

Exemplu: Calculaţi adresa fizică şi conţinutul locaţiei respective de memorie după execuţia următoarelor instrucţiuni, presupunând DS=1470h:

```
mov AL, 50h
mov [4320h], AL
```

Adr.Fizică = 18A20h ; [18A20h] = 50h.

B. Operanzii cu adresare indirectă utilizează regiştri pentru a indica adrese din memorie. Adresarea indirectă este folosită în manipularea dinamică a datelor, deoarece valorile din regiştri se pot modifica. În cazul microprocesoarelor 8086 numai patru regiştri pot fi folosiţi în adresarea indirectă: regiştrii de bază BX şi BP şi regiştrii index SI şi DI. Regiştrii de bază sau index pot fi folosiţi împreună sau separat, cu sau fără specificarea unui deplasament. Forma generală pentru accesarea indirectă a unui operand de memorie este:

[BX | BP] + [DI | SI] + [deplasament 8/16 biţi]

adică adunând următoarele trei elemente, sau numai unele dintre ele:
- conţinutul unuia dintre regiştrii BX sau BP
- conţinutul unuia dintre regiştrii DI sau SI
- un deplasament.

Rezultă astfel următoarele moduri de adresare la memorie:
- *directă* – atunci când apare numai constanta
- *bazată* – atunci cand în calcul apare un registru de bază
- *indexată* – atunci când în calcul apare un registru index

sau combinaţii ale acestora.
Dacă BX este folosit ca registru de bază sau dacă nu este specificat nici un registru de bază, la calculul adresei efective a unui operand cu adresare indirectă DS va fi folosit ca registru segment implicit. Dacă BP este folosit oriunde în calculul adresei operandului, segmentul implicit este SS.

Se permit diverse modalităţi de a specifica operanzi cu adresare indirectă, folosind orice operator care indică adunarea (+, [ ], .). De exemplu următoarele moduri de specificare sunt echivalente:

```
table [BX][DI] + 6
6 + table [BX+DI]
[table+BX+DI] + 6
table [BX+6][DI]
```

Când se utilizează modul de adresare bazat-indexat, unul dintre regiştri trebuie să fie registru de bază, iar celălalt să fie registru index. Următoarele instrucţiuni sunt **incorecte:**

```
mov AX, table [BX][BP]        ; doi regiştri de bază!
mov AX, table [SI][DI]        ; doi regiştri index!
```

Pentru adresarea indirectă, esenţială este specificarea între paranteze drepte a cel puţin unuia dintre elementele componente ale formei prezentate.

Codificarea instrucţiunilor se realizează în funcţie de modul de adresare folosit. Formatul unei instrucţiuni este de forma:

Cod	d	w	octet - mod de adresare	depl L	depl H

cod – este codul mnemonicii (operaţiei)
d – indică dacă locaţia de memorie este sursă sau destinaţie
d=0 destinaţie, d=1 sursă
w – indică dacă operaţia se face pe octet sau pe cuvânt
w=0 octet, w=1 cuvânt
octetul mod de adresare – este alcătuit din 3 părţi distincte:

7	6	5	4	3	2	1	0
mod			reg		r/m		

mod =
00 – adresare cu memoria, fără deplasament
01 – adresare cu memoria, deplasament pe un octet
10 – adresare cu memoria, deplasament pe 2 octeţi
11 – adresare registru

reg – indică registrul operand în instrucţiune

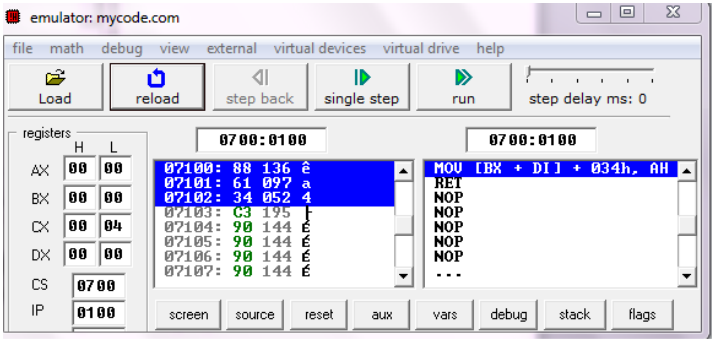
r/m – indică modul de calcul al adresei efective
(registrul segment implicit)

reg	w=0	w=1
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

Exemplu: `mov [BX+DI+34h], AH ;`

100010	0	0	01	100	001	0011	0100
cod “**mov**”	d	w	mod	reg (AH)	r/m (BX+DI+depl8)	3	4

Folosind simulatorul EMU8086 verificati exemplul de mai sus.



Exemplu: `mov [bx+di+34h], ah ;`

8	8		6	1		3	4
100010	0	0	01	100	001	0011	0100
cod “**mov**”	d	w	mod	reg (AH)	r/m (bx+di+depl8)	3	4

Fig.2.2. Codificarea instrucţiunii mov [BX+DI+34h], AH

Analizati manual dar şi verificaţi codul următoarelor instrucţiuni :

```
mov [BX+DI+1234h], AH ;
mov [BX+DI+1234h], AX ;
```

2.5 Exerciţii şi teme

1. Daţi exemplu de 3 adrese logice ce se pot scrie pt adresa fizică 23456h.

2. Să se calculeze adresa fizică ce corespunde adresei logice 89ABh : 89ABh.

3. Să se calculeze componenta offset corespunzătoare adresei fizice 10000h dacă se dă componenta segment 1000h. Dar dacă adresa fizică este 1FFFFh ?

4. Să se calculeze componenta segment corespunzătoare adresei fizice 0ABC10h dacă se dă componenta offset 600h.

5. Verificaţi dacă adresa fizică 31FFFh aparţine segmentului care are componenta segment 2200h. Dar dacă adresa fizică este 21000h?

6. Conţinutul căror locaţii de memorie este mutat în AX în instrucţiunile următoare? Ce mod de adresare este folosit? Se considera BX=12ABh, BP=1300h, SI=1A2Bh, DI=340Ch, DS=2100h, SS=3F00h, CS=5A00h.

```
mov AX, [BX+3]
mov AX, 4[BX+SI]
mov AX, [SI]+10h
mov AX, 5[BP]
mov AX, [DI+8]
mov AX,2345h
```

7. Ştiind CS=1200h, SS=3F00h, IP=2000h şi SP=2002h, specificaţi adresele logice şi fizice pentru a) elementul din vârful stivei şi b) instrucţiunea curentă.

8. Pentru SS=3F00h, CS=5A00h, ES=1400h, DS=1200h, SI=1234h, DI=340Ch, AX=2ABCh şi BX=1A3Bh, BP=1300h, menţionaţi tipul de adresare folosit şi specificaţi adresa locaţiei de memorie accesată de fiecare dintre instrucţiunile următoare, folosind modelul:

```
mov AX, [BX]
```

- s-a folosit adresarea bazată
- primul operand este registrul AX
- al doilea operand este perechea de octeţi (**cuvântul**) din memoria principală, din segmentul curent de date (specificat de DS), aflat la offset-ul conţinut în registrul BX
- mai precis, operandul este cuvântul din memorie, de la adresa 1200h:1A3Bh (partea LOW) şi 1200h:1A3Ch (partea HIGH)

```
mov [DI], AX
mov [SI], AL
mov AX, [SI+BX]
mov AL, [BX] [SI]
mov AX, DS: [BP+2]
```

9. Precizaţi dacă următoarele instrucţiuni sunt corecte şi justificaţi răspunsul:

```
mov AX, [BP+BX]
mov AL, [BX]
mov AX, [BP+SI]
mov AX, [SI+DI]
mov [SI+DI], AX
mov [SI+BP], AX
mov AX, [BP+5]
```

```
mov AX, 7[BP]
mov AX, [6]
```

10. Analizaţi secvenţa de mai jos şi scrieţi atât rezultatul obţinut cât şi codificarea pentru următoarele secvenţe de instrucţiuni; comparaţi rezultatele obţinute între ele, specificând efectul fiecărei instrucţiuni în parte:

```
org 100h
.data ; în zona de date se definesc cele 2 variabile de mai jos
sir1 db 1,2,3,4,5,6
sir2 dw 7,8,9,10,11,12,13,14,15

.code
mov al, sir1 [2];   _______          mov bl, sir1 [3]; ________

mov ax, sir2 [2];  ________          mov bx, sir2 [3]; ________

mov sir1 [2], ah ; _______          mov sir1 [3], bh; ________

mov sir2 [2], ax ; ________          mov sir2 [3], bx; ________
```

11. Fie o variabilă definită în memorie *sir db 0,1,2,3,4,5,6,7,8,9.* Scrieţi câte un exemplu de instrucţiune *mov* folosind adresare a) bazată, b) indexată şi c) bazată-indexată astfel încât să se depună în registrul AL al 5-lea element al şirului.

12. Precizaţi dacă următoarele instrucţiuni sunt corecte şi justificaţi răspunsul:

```
mov BX, [AX]
mov AX, [BX]
mov AX, [CX+4]
mov AL, sir[3+BX]
mov AX, sir[3+BL]
mov AL, sir [CH+5]
mov BL, sir 2[BH]
mov AX, [DX]
```

13. Folosind regulile specificate în material, specificaţi modul de codificare al următoarelor instrucţiuni:

```
mov [BX+DI+1234h], AL
mov [BX+DI+1234h], AX
mov AX, [BX+DI+1234h]
mov BX,CX
mov BH,CH
mov BL,CH
mov BP,SP
mov [BX+10],5678h
```

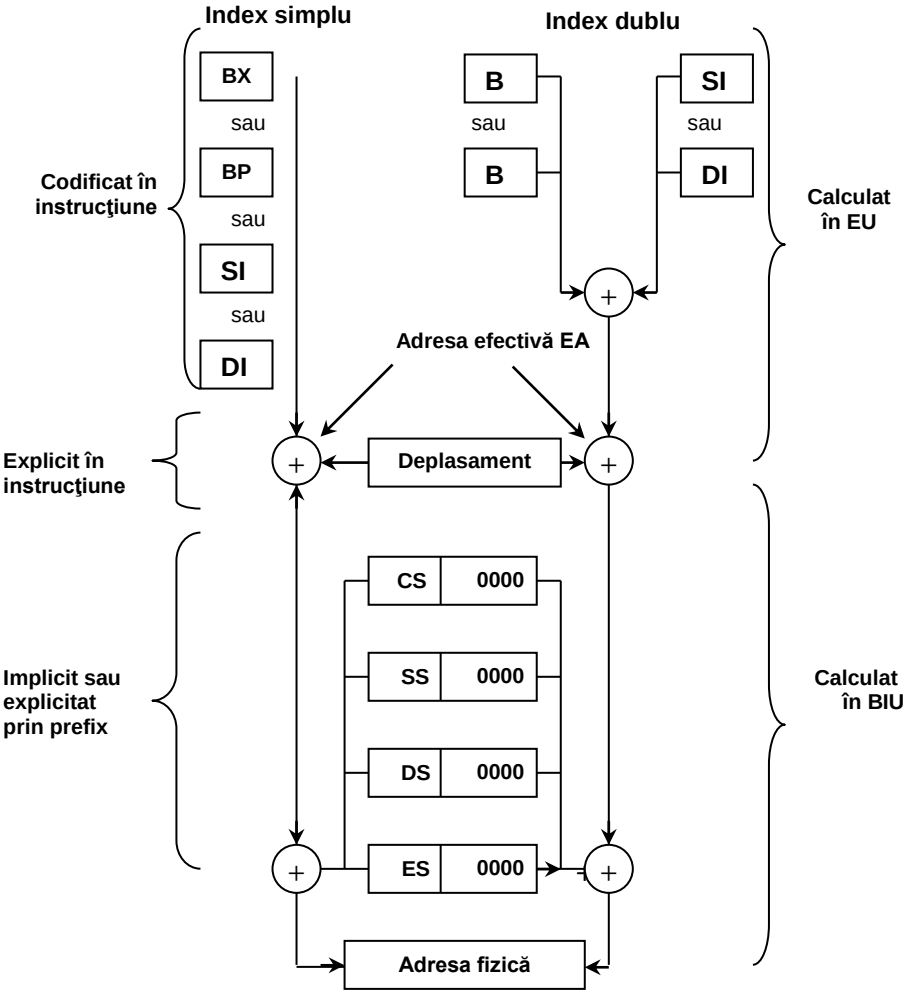


Fig. 2.3. Variante de calcul ale adresei fizice de memorie

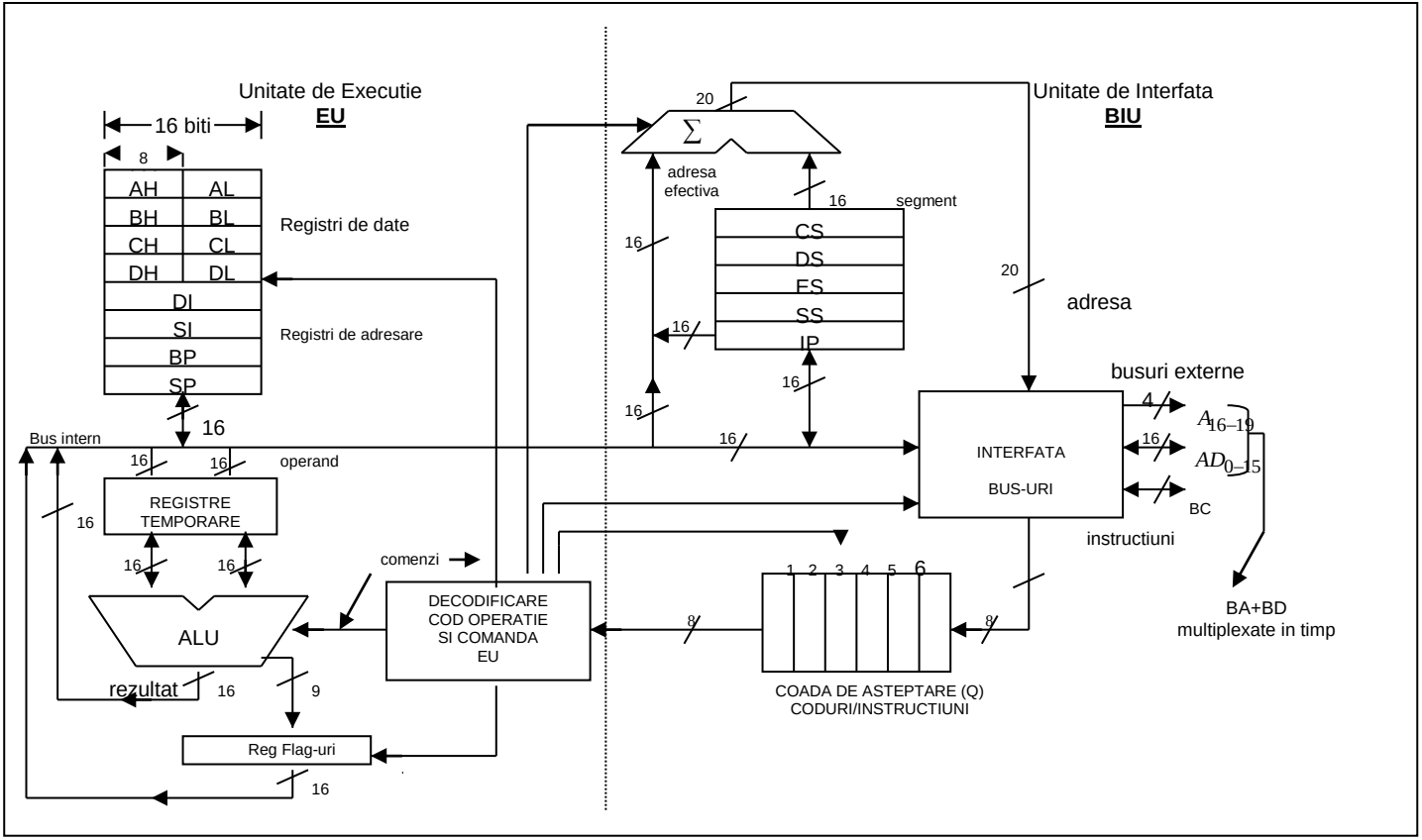


Fig. 2.3 Schema bloc internă simplificată a microprocesorului 8086