Arhitectura microprocesorului Intel 8086

Scopul lucrării

a) Studiul arhitecturii interne a microprocesorului Intel 8086.
b) Studiul resurselor interne (registre) si externe (memoria principala) ale microprocesorului I 8086.
c) Programarea microprocesorului Intel 8086.

1. Microprocesorul Intel 8086

1.1. Sisteme cu microprocesor Intel 8086

Microprocesorul este o unitate centrala de prelucrare (UCP) realizata intr-un singur circuit integrat. Un sistem digital de prelucrare realizat cu ajutorul unui microprocesor este numit *microcalculator*. Microcalculatorul are trei functii principale: prelucrarea informatiilor in UCP, stocarea informatiilor in memorie si transferul informatiilor in interior si cu mediul exterior.

Prin intermediul unor interfete, numite *porturi de intrare/iesire* (IO), microcalculatorul transfera informatii cu elementele mediului exterior, numite echipamentele periferice. Echipamente periferice sunt: tastatura, monitor, imprimanta, disc hard, discheta, cititor de CD-ROM, etc.

Microprocesorul Intel 8086 este o unitate centrala de prelucrare care este formata din 2 componente:

1. Unitatea de executie (UE) decodifica instructiunile numerice, da comenzi interne pentru efectuarea calculelor si comenzi externe catre cea de-a doua unitate. UE contine 8 locatii de memorie interna numite *registre de uz general*. Registrele ofera o *capacitate de memorare mica* dar si un *acces (citire sau scriere) foarte rapid*.

Pentru a stoca o cantitate mai mare de date (codurile numerice ale instructiunilor si variabilele programelor) este necesara conectarea microprocesorului cu o *memorie de capacitate mare*, numita *memorie principala* (MP). Desigur ca *accesul la MP este mult mai lent*.

2. Unitatea de interfata cu bus-urile (UI) calculeaza adresele MP si IO, transfera datele intre UE si MP sau intre UE si I/O, si transfera catre UE codurile numerice ale instructiunilor citite din MP.

Pentru *a identifica in mod unic* fiecare dintre locatiile MP si IO, este necesara asocierea unor *referinte numerice* numite *adrese*. De aceea UI este responsabila cu *generarea adreselor* catre MP.

Microprocesorul Intel 8086 lucreaza cu *date de 16 biti*, numite *cuvinte* de date. Transferurile intre UE si UI sau intre microprocesor si MP sau IO se fac in general sub forma de cuvinte de date de 16 biti. De aceea bus-ul intern prin care comunica UE si UI al microprocesorului Intel 8086 este de 16 biti.

Pentru compatibilitate cu microprocesoarele care lucrau cu date de 8 biti, si Intel 8086 poate transfera valori sub forma unor *octeti* (date de 8 biti). De aceea locatiile MP si ale IO sunt octeti.

Microprocesorul Intel 8086 poate lucra cu maximum 1 M octeti de MP, adica poate genera cel mult 1 M de adrese distincte. Deoarece $1M = 2^{16}$, inseamna ca *adresele* la microprocesorul Intel 8086 sunt *reprezentabile cu 20 de biti.*

In figura 1 este prezentata schema bloc a unui *sistem cu microprocesor I8086.*

Microprocesorul Intel 8086 comunica cu exteriorul (MP si IO) prin 3 bus-uri sau magistrale:
- bus-ul de date (BD), care are 16 biti;
- bus-ul de adrese (BA), care are 20 biti;
- bus-ul de comenzi (BC).

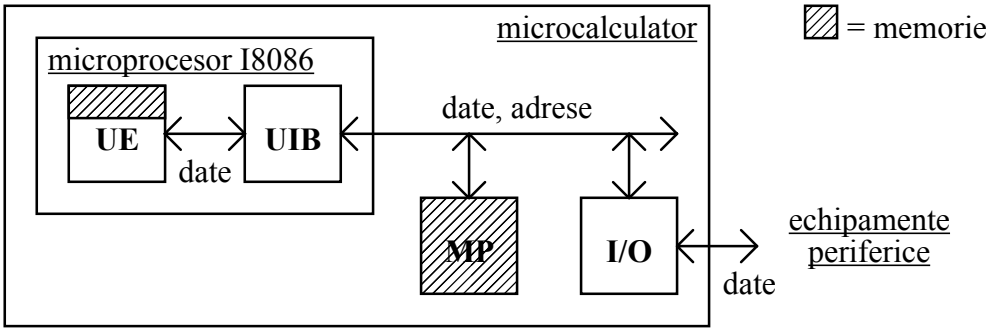

Fig.1. Schema bloc a unui sistem cu microprocesor I8086.

1.2. Arhitectura microprocesorului Intel 8086

Structura microprocesorului Intel 8086 este de tip "pipe-line" (prezinta *paralelism temporal*) permitand efectuarea in acelasi timp a doua operatii diferite de catre cele doua unitati diferite ale sale - UE decodifica instructiunile si efectueaza calculele, in timp ce UI calculeaza adresele si efectueaza transferurile.

Astfel, cele doua unitati ce compun UCP efectueaza autonom *secvente de operatii proprii*, transferindu-si in acelasi timp informatii. Secventele de operatii efectuate de cele doua unitati ale microprocesorului pentru a executa instructiunile sunt numite *cicluri de instructiune*, pentru UE si *cicluri masina de bus*, pentru UI.

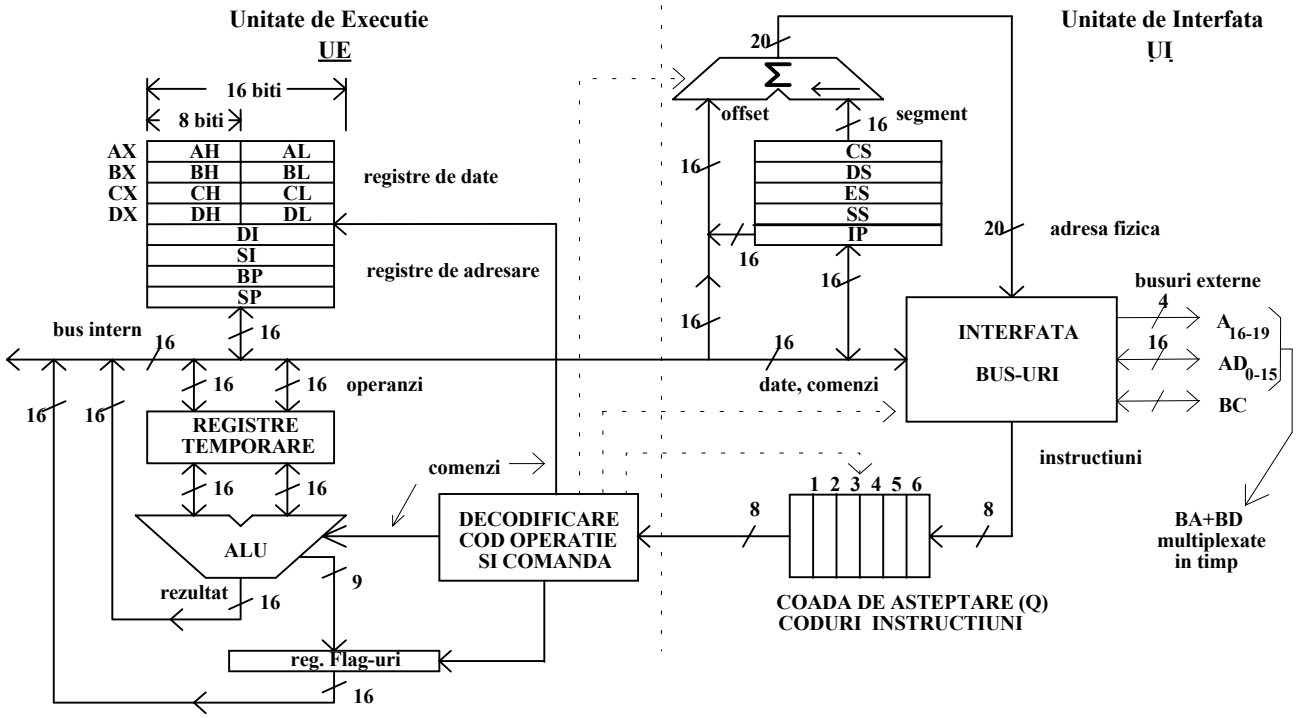

Fig.2 Arhitectura interna a microprocesorului Intel 8086

Unitatea de executie (UE) se compune din:

1. Unitatea aritmetica-logica (ALU), care executa operatii aritmetice, logice, deplasari si rotatii.

2. Registrele temporare (RT), care preiau operanzii de pe bus-ul intern si ii ofera unitatii ALU. Impreuna cu ALU formeaza un *automat RALU.*

3. Registrul de flag-uri F (biti indicatori de stare a ultimei operatii ALU), actualizat de catre ALU.

4. Blocul de comanda, care:

- decodifica codul instructiunii curente (preluat din coada de instructiuni Q);
- da comenzi interne catre celelalte blocuri ale UE pentru:
  - calculul adreselor efective ale operanzilor din MP sau IO (daca este cazul),
  - executia operatiei (ALU, transfer, etc.);
- da comenzi externe (informatii / cereri) catre UI pentru:
  - calculul adreselor fizice a operandului din MP sau IO (daca este cazul),
  - transferul operanzilor dinspre/catre MP sau IO (daca e cazul),
  - transferul (citirea) operanzilor de tip imediat din coada de asteptare a UI (daca e cazul),
  - calculul adresei urmatoarei instructiuni.

Se observa ca unitatea de executie este *complet separata de exterior*, toate sarcinile privind transferul cu exteriorul revenind unitatii de interfata.

Unitatea de interfata cu bus-urile (UI) se compune din:

1. Blocul de interfata intre bus-uri, care face toate transferurile la cererea UE:
- cicluri de scriere in MP sau IO (dinspre UCP catre exterior);
- cicluri de citire din MP sau IO (dinspre exterior catre UCP).

2. Coada de asteptare a codurilor de instructiune (Q), care:
- este incarcata cu coduri de instructiune (ciclu FETCH) de catre blocul de interfata atunci cind UE nu cere transferuri de date;
- este inactiva (UCP executa cicluri inactive de bus) daca este plina si nu se cer transferuri;
- este stearsa complet (resetata) daca instructiunea curenta este de salt.

3. Blocul de calcul al adreselor fizice incluzand:
- registrele segment, care contin componenta sement a adresei locatiei MP accesate;
- registrul indicator al instructiunii curente (IP), care contine componenta offset a adresei instructiunii curente;
- unitatea de deplasare-adunare pentru calculul adresei fizice din componentele segment si offset.

2. Registrele microprocesorului Intel 8086

Registrele *de capacitate 16 biti* ale microprocesorului Intel 8086 pot fi clasificate din punct de vedere al rolului pe care il au in executia instructiunilor in 4 grupuri:
- registrele generale: AX, BX, CX, DX, SP, BP, SI, DI;
- registrele segment: CS, DS, ES, SS;
- registrul indicator al adresei instructiunii curente: IP;
- registrul de flag-uri: F.

2.1. Registrele generale

Registrele generale se pot imparti in doua seturi de registre:
- registre de date: AX, BX, CX, DX;
- registre de adresare: SP, BP, SI, DI;

1. Registrele de date se deosebesc prin faptul ca jumatatile (*de capacitate 8 biti*) lor pot fi accesate (citite sau scrise) separat. Aceasta inseamna ca fiecare registru de date poate fi folosit ca un registru de 16 biti sau ca 2 registre de 8 biti :
- AX (*acumulator*) de 16 biti, poate fi accesat ca AH si AL, ambele de 8 biti;
- BX (*baza in adresarea datelor*) de 16 biti, poate fi accesat ca BH si BL, ambele de 8 biti;
- CX (*contor*) de 16 biti, poate fi accesat ca CH si CL, ambele de 8 biti;
- DX (*date*) de 16 biti, poate fi accesat ca DH si DL, ambele de 8 biti.

Registrele de date sunt utilizate in majoritatea instructiunilor aritmetice si logice. Majoritatea instructiunilor aritmetice utilizeaza in acelasi mod toate registrele.

Exista si instructiuni aritmetice pentru care anumite registre generale au *intrebuintari speciale*, prezentate in continuare:

AX - operatii de intrare/iesire pe 16 biti, implicit in inmultiri si impartiri pe 16 biti;
AL - operatii de intrare/iesire pe 8 biti, implicit in translatii, aritmetica BCD, inmultiri si impartiri pe 8 biti;
AH - implicit in inmultiri si impartiri pe 8 biti;
BX - operatii cu memoria - adresare indirecta, implicit in translatii;
CX - implicit in operatii cu siruri sau bucle;
CL - operatii de deplasare sau rotatie cu mai mult de 1 pozitie;
DX - operatii de intrare/iesire - adresare indirecta, implicit in inmultiri si impartiri pe 16 biti.

2. Registrele de adresare, de 16 biti, se impart la rindul lor in:
- *registre indicatoare de adresa in stiva (pointer):*
  - SP (Stack Pointer) - contine adresa curenta a varfului stivei,
  - BP (Base Pointer) - implicit contine adresa de baza pentru adresarea indirecta a stivei ;
- *registre indicatoare de adresa pentru siruri (index)*:
  - DI (Destination Index) - implicit contine adresa curenta pentru sirul destinatie,
  - SI (Source Index) - implicit contine adresa curenta pentru sirul sursa.

Registrele de adresare pot fi utilizate si pentru date in anumite instructiuni aritmetice si logice.

Registrele pointer contin componente offset ale adreselor din stiva (adresele relative in segmentul de stiva curent). Registrul *BP poate fi utilizat si pentru adresarea in cadrul altor segmente.*

Registrele index contin componente offset ale adreselor variabilelor (adresele relative in segmentul de date curent). Ele sunt utilizate ca registre de adresare in instructiunile de transfer sau prelucrare de siruri de octeti (caractere). In acest ultim caz SI contine adresa relativa curenta a sirului destinatie in cadrul segmentului de date curent (DS), iar DI contine adresa relativa curenta a sirului sursa in cadrul segmentului de date suplimentar (ES).

2.2. Registrele segment

Spatiul de memorie de 1 Moctet este impartit in *segmente logice* de lungime 64K octeti. UCP Intel 8086 are acces la patru segmente deodata prin intermediul a patru registre segment.

Cele 4 registre segment sunt:
- CS (Cod Segment) - contine componenta segment a adreselor codului (instructiunilor programului);
- DS (Data Segment) - contine componenta segment a adreselor variabilelor (segment date curent);
- ES (Extra Segment) - contine componenta segment a adreselor variabilelor (segment suplimentar);

- SS (Stack Segment) - contine componenta segment a adreselor datelor din segmentul stiva.

Instructiunea care urmeaza sa se execute se gaseste in segmentul a carui adresa se afla in registrul CS (Cod Segment), la adresa relativa continuta in registrul IP.

Continutul registrului DS defineste segmentul de date curent. Toate referirile la datele din memorie, cu exceptia celor prin registrele BP si SP sau registrul DI in instructiunile pentru siruri, utilizeaza *in mod implicit* segmentul referit de registrul DS.

Continutul registrului ES defineste segmentul de date suplimentar. Referirile la date in instructiunile pentru siruri utilizeaza *in mod implicit* segmentul referit de registrul ES.

Continutul registrului SS defineste segmentul curent al stivei. Toate referirile la datele din memorie prin registrele BP si SP utilizeaza *in mod implicit* segmentul referit de registrul SS.

2.3. Alte registre

Registrul indicator al adresei instructiunii curente IP (Instruction Pointer) este un registru de 16 biti care contine componenta offset a adresei instructiunii in segmentul de cod curent. *Programele nu au acces direct la IP*, dar exista instructiuni care il modifica si il incarca sau il descarca prin stiva.

Registrul de flag-uri F cuprinde bitii indicatori de stare si control numiti si *biti indicatori de conditii*, fanioane sau *flag-uri*. Indicatorii de conditii sunt utilizati pentru a memora informatii referitoare la rezultatul unor operatii aritmetice si logice (OF,SF,ZF,AF,PF,CF) si pentru memorarea unor informatii de control al microprocesorului (TF,DF,IF).

Cei 9 biti mentionati sunt plasati in registrul F ca in figura 3.

X	**X**	**X**	**X**	**O**	**D**	**I**	**T**	**S**	**Z**	**X**	**A**	**X**	**P**	**X**	**C**

Fig. 3. Continutul registrului indicatorilor de stare si control

Semnificatiile flag-urilor sint urmatoarele:

- CF (Carry Flag) - *indicator de transport* - reflecta transportul in exterior al bitului cel mai semnificativ al rezultatului operatiilor aritmetice. Astfel, acest indicator poate fi folosit in cazul operatiilor in dubla precizie. Valoarea CF = 1 semnifica fie transport la adunare fie imprumut la scadere. De asemenea, indicatorul CF este modificat si de instructiunile de deplasare si rotatie.

- PF (Parity Flag) - *indicator de paritate* - este 1 daca rezultatul are paritate para (contine un numar par de biti 1). Acest indicator este folosit de instructiunile de aritmetica zecimala.

- AF (Auxiliary Carry Flag) - indicator de transport auxiliar - este 1 daca a fost transport de la jumatatea de octet inferioara la jumatatea de octate superioara (de la bitul 3 la bitul 4). Acest indicator este folosit de instructiunile de aritmetica zecimala.

- ZF (Zero Flag) - indicatorul de zero - este 1 daca rezultatul operatiei a fost zero.

- SF (Sign Flag) - *indicatorul de semn* - este 1 daca cel mai semnificativ bit al rezultatului (MSb) este 1, adica in reprezentarea numerelor in complement fata de 2 (C2) rezultatul este negativ (are semn -).

- OF (Overflow Flag) - *indicatorul de depasire aritmetica* (a gamei de valori posibil de reprezentat) - este 1 daca dimensiunea rezultatului depaseste capacitatea locatiei de destinatie si a fost pierdut un bit (indica la valorile cu semn faptul ca se "altereaza" semnul).

- IF (Interrupt Flag) - *indicatorul de validare a intreruperilor* - prin valoarea lui 1 permite UCP sa recunoasca cererile de intrerupere externe mascabile. Prin valoarea 0 intreruperile externe mascabile vor fi invalidate. Acest indicator nu afecteaza intreruperile interne sau pe cele externe nemascabile.

3. Memoria principala a unui sistem cu microprocesor

Microprocesorul INTEL 8086 poate adresa un *spatiu de memorie principala* (MP) de 1Moctet. Instructiunile si datele, pentru a oferi o utilizare eficienta a memoriei, pot fi stocate in memorie fara sa conteze alinierea lor.

Conform *conventiei INTEL, datele formate din mai multi octeti (multioctet)* sunt intotdeauna memorate cu octetul cel mai semnificativ *(MSB) la locatia de adresa cea mai mare*. Asadar octetul cel mai putin semnificativ *(LSB) este memorat la adresa cea mai mica* din grupul de octeti ai unei date multioctet.

3.1. Gestiunea memoriei principale. Segmentarea

Microprocesorul Intel 8086 *vede* memoria organizata ca un grup de segmente. Un *segment* este un bloc de memorie, constituit din locatii de memorie contigue, de dimensiune 64 Kocteti.

Fiecare segment poate fi accesat (poate fi citit sau scris) in mod independent. Pentru aceasta este necesar ca fiecare segment sa poata fi adresat independent. De aceea un segment are asociata o *adresa de baza*, care este *adresa locatiei la care incepe segmentul*. Aceasta adresa este multiplu de 16.

Segmentele pot fi *adiacente, disjuncte, partial suprapuse* sau *suprapuse complet* (ca in figura 4). O locatie fizica poate sa apartina unuia sau mai multor segmente. Fiecare program (aplicatie) defineste si utilizeaza segmentele diferit.

Spatiul de memorie accesibil la un moment dat este de 64 K octeti pentru cod (prin intermediul CS), 64 K octeti pentru stiva (prin SS), 128 K octeti pentru date (prin DS si ES) (vezi figura 4).

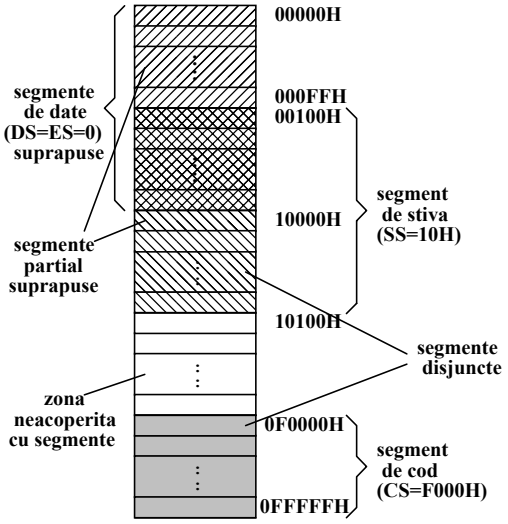

Fig. 4. Exemplu de organizare a memoriei in segmente

3.2. Generarea adresei fizice

Fiecare locatie de memorie are o adresa propriu-zisa, numita *adresa fizica*, pe care microprocesorul Intel 8086 o calculeaza din cele doua componente, segment si offset, ale *adresei logice*.

Adresa fizica este o valoare de *20 biti* care identifica unic o locatie din spatiul de adresare. Adresa fizica poate fi in domeniul 0H ... 0FFFFFH.

Pentru ca programele sa fie relocabile, microprocesorul foloseste insa o adresa logica pentru a calcula adresa fizica.

Adresa logica consta dintr-o componenta *segment* de *16 biti* si o componenta *offset* de *16 biti*. Din componenta segment microprocesorul poate calcula *adresa de baza (de inceput) a segmentului* prin inmultirea cu *10 H = 10000 B*.

Adresa fizica se calculeaza adunand la adresa de baza a segmentului componenta offset. De aceea componenta offset se mai numeste si *deplasare, adresa relativa sau adresa efectiva*, iar adresa fizica se mai numeste si *adresa absoluta*.

Notatia consacrata pentru *adresa logica* este urmatoarea:

segment : offset

Calculul *adresei fizice* se face cu formula:

*segment * 10 H + offset*

unde: *segment * 10 H* este adresa de baza a segmentului,
offset este adresa relativa la baza segmentului:

Deoarece inmultirea cu 10 H = 10000 B este echivalenta cu deplasarea cu o cifra hexazecimala la stanga, respectiv cu deplasarea cu 4 cifre binare (biti) la stanga, rezulta ca adresa fizica se calculeaza prin deplasarea cu 4 biti la stanga a segmentului si adunarea cu offset-ul, asa cum este ilustrat in exemplul din figura 5.

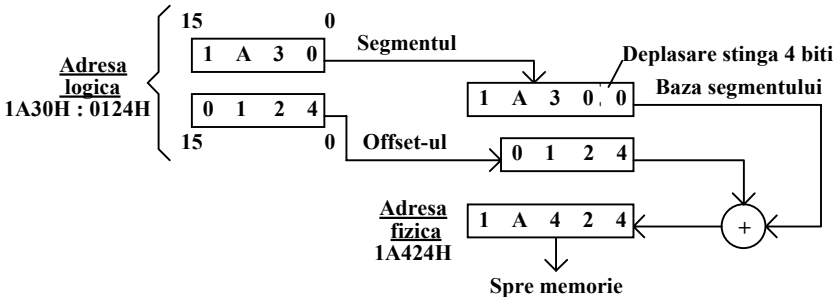

Fig. 5. Calculul adresei fizice din adresa logica

Pentru orice locatie de memorie, *adresa de baza a segmentului* este adresa primului octet al segmentului care contine locatia, iar *adresa relativa* este distanta in octeti de la inceputul segmentului pina la locatia respectiva (vezi figura 6).

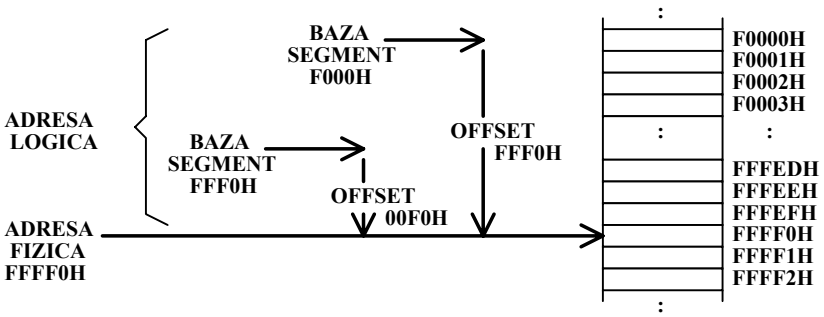

Fig. 6. Adresa logica (16 biti + 16 biti) si adresa fizica (20 biti)

Aceeasi locatie poate fi accesata (citita sau scrisa) prin diferite adrese logice. De exemplu, din adresa logica *F000H : FFF0H* se calculeaza adresa fizica *F000H * 10H + FFF0H = FFFF0H*, iar din adresa logica *FFF0H : 00F0H* se calculeaza adresa fizica *FFF0H *10H + F0H = FFFF0H.*

Unitatea de interfata cu magistrala (UI) *obtine adresa logica a unei locatii de memorie diferit in functie de modul de adresare* a memoriei.

Structura pe segmente a memoriei UCP Intel 8086 face posibila *scrierea unor programe care sunt independente de pozitia lor in memorie*, adica sint *relocabile* dinamic.

Pentru ca un program sa fie relocabil dinamic trebuie sa fie scris astfel incat sa nu altereze registrele sale segment si sa nu faca transferuri directe la o locatie in afara segmentului de cod. Aceasta permite programului sa fie mutat oriunde in memoria disponibila, atita timp cit registrele segment sunt actualizate cu noua adresa de baza.

4. Programarea microprocesorului Intel 8086

Microprocesorul Intel 8086 *efectueaza prelucrari sub comanda unui program* numit si *program de aplicatie. Programul* este o secventa de instructiuni aflata in memoria principala.

Instructiunea este o comanda elementara data UCP in vederea executarii:
- operatiilor asupra unor date (transferuri, aritmetice-logice, deplasari, rotatii, etc.);
- comenzilor in vederea selectarii instructiunilor urmatoare, sau trecerii UCP in stari speciale (ex. asteptarea producerii unor evenimente/semnale externe numite intreruperi);
- unor operatii auxiliare.

Instructiunile sint *codificate in cod binar*, ocupind in memorie 1-6 octeti.

Codul instructiunii microprocesorului Intel 8086 este format din:
- *codul operatiei* (primii 1-2 octeti), care specifica (codifica):
  - tipul operatiei;
  - tipul operanzilor (8 sau 16 biti, etc.);
  - sursa operanzilor (interna sau externa);
  - destinatia rezultatelor operatiilor ALU;
  - modul de calcul al EA (daca este cazul).
- *operanzii de tip imediat* (daca exista):
  - date;
  - adrese.

Exemplu de codificare:

4.1. Etapele dezvoltarii unui program pentru microprocesorul Intel 8086

Desi limbajele de nivel inalt sunt preferate in general datorita posibilitatilor de structurare a programelor, in cazul sistemelor dedicate realizate cu microprocesor, este util si uneori necesar ca o parte sau uneori intregul program sa fie scris in limbaj de asamblare.

Principalul avantaj al programelor scrise in limbaj de asamblare este *viteza maxima de executie*. Acesta deriva din faptul ca in *limbaj de asamblare* operatiile limbajului (instructiunile) sunt cele ale microprocesorului.

O alta caracteristica a limbajului de asamblare este faptul ca referintele simbolice (etichete, variabile) sunt cele ale unor elemente din memorie (corespund unor adrese).

Procesul prin care se ajunge de la *punerea unei probleme* pe care trebuie sa o rezolve microprocesorul la crearea unui program care sa indeplineasca corect cerintele acelei probleme se numeste *dezvoltare* a acelui program.

Etapele dezvoltarii unui program pentru microprocesorul Intel 8086 sunt:

1. Conceperea programului in limbaj de asamblare.
2. Editarea programului cu ajutorul unui editor de texte, obtinandu-se textul sursa.
3. Asamblarea textului sursa, obtinandu-se codul obiect.
4. Corectarea eventualelor erori de asamblare, reluandu-se etapa 2 daca este cazul.
5. Editarea de legaturi a codului obiect, obtinandu-se codul executabil.
6. Corectarea eventualelor erori de editare de legaturi, reluandu-se etapa 2 daca este cazul.
7. Lansarea in executie a programului, comparandu-se efectul cu cel dorit.
8. Corectarea eventualelor erori conceptie, reluandu-se etapa 1 daca este cazul.

Asamblorul este un sistem software (SW) care asista programatorul in elaborarea programelor in cod masina. *Programul asamblor converteste reprezentarile simbolice* ale instructiunilor *in configuratii binare* obtinind echivalentele in *cod masina* ale instructiunilor.

Asamblorul este de fapt un compilator simplificat pentru programe sursa scrise in limbaj de asamblare. Asamblarea se face de obicei in doi pasi:

- la prima parcurgere a textului *se culeg toate referintele* simbolice (denumirile) si se introduc *intr-un tabel de simboluri*;
- la a doua parcurgere a textului sursa se face *traducerea folosind informatiile din tabel.*

Informatiile simbolice intalnite de asamblor in *textul sursa* al unei sectiuni pot fi:

- *elemente absolute* (coduri de operatii, constante, nume simbolice ale instructiunilor din sectiunile absolute);
- *informatii relocabile*, a caror valoare depinde de pozitia lor relativa in sectiune si care poate sa difere de la o executie la alta, in functie de adresa de incarcare a sectiunii in memorie; valoarea unui element relocabil este data de pozitia definitiei sale in textul sursa, fata de inceputul sectiunii;
- *referinte externe* (simboluri *globale*) - nume simbolice definite intr-o sectiune si referite in late sectiuni si a caror valoare se stabileste numei in momentul legarii sectiunilor intr-un program unic.

Rezultatul asamblarii este un *text in forma binara (cod obiect)* in care se *pastreaza in forma initiala numai referintele externe*. In plus se furnizeaza informatii indicind locul referintelor relocabile pentru ca in momentul incarcarii valorile lor sa devina referinte absolute.

Combinarea sectiunilor se face de catre un program *editor de legaturi* prin *rezolvarea referintelor externe* (adica inlocuirea numelor cu adrese din memorie) si adaugarea eventual a rutinelor din bibliotecile standard (care sint pastrate tot intr-o forma relocabila).

Programul rezultat este deja pus intr-o *forma interna*, *direct executabila*, el trebuind eventual numai relocatat de catre un *program locator*. Locatorul este *apelat in momentul punerii in executie* a programului creat.

Executia poate fi obtinuta fie ca o comanda data sistemului de operare din linia de comanda, fie prin lansarea in executie dintr-un mediu specializat. Un astfel de mediu specializat este *depanatorul simbolic*, care permite executia instructiune cu instructiune cu urmarirea resurselor (registre, flag-uri, memorie, adrese, stiva, etc.).

Exemple de programe utilizate in loborator pentru dezvoltarea unui program:
- EDIT = editor de texte pentru sistemul de operare MS-DOS,
- BC = mediu de dezvoltare pentru programe scrise in C, C ++, utilizabil ca editor de texte,
- TURBO = mediu de dezvoltare pentru programe scrise in Pascal, utilizabil ca editor de texte,
- TASM = asamblor,
- TLINK = editor de legaturi,
- TD = depanator simbolic.

4.2. Depanatorul simbolic Turbo Debugger

Depanatorul simbolic Turbo Debugger (TD) este o componenta a mediilor de dezvoltare Pascal si C in variantele Turbo sau Borland actuale. El permite vizualizarea resurselor unui PC (Personal Computer), care este un sistem echipat cu microprocesor, in general din familia Intel 80x86.

Resursele sunt interne (registrele microprocesorului) sau externe (valorile unor locatii de memorie principala). Ele pot fi urmarite in timpul executiei unui program in cod masina sau in limbaj de nivel inalt (Pascal, C), instructiune cu instructiune sau dupa executia unei secvente de instructiuni.

Pe ecran resursele apar in 5 subferestre, dupa cum urmeaza:

- *subfereastra codului programului* (continutul memoriei principale asociate segmentului de cod), astfel:
  - in stanga adresele, in forma *segment : offset,*
  - in mijloc codul masina numeric,
  - in dreapta codul simbolic (instructiunile in limbaj masina),
- *subfereastra registrilor microprocesorului*,
- *subfereastra flag-urilor*,
- *subfereastra memoriei principale* (continutul memoriei principale asociate segmentului specificat), astfel:
  - in stanga adresele, in forma *segment : offset,*
  - in mijloc codurile numerice,
  - in dreapta codurile ASCII asociate,
- *subfereastra stivei* (continutul memoriei principale asociate segmentului de stiva).

= **F**ile **E**dit **V**iew **R**un **B**reakpoints **D**ata **O**ptions **W**indow **H**elp

```
cs:0000 > B83454      mov   ax,5434        ax  0000   c = 0
cs:0003   8ED8        mov   ds,ax          bx  0000   z = 0
cs:0005   A10000      mov   ax,[0000]      cx  0000   s = 0
cs:0008   8B1E0200    mov   bx,[0002]      dx  0000   o = 0
cs:000C   03060400    add   ax,[0004]      si  0000   p = 0
cs:0010   A30800      mov   [0008],ax      di  0000   a = 0
cs:0013   131E0600    adc   bx,[0006]      bp  0000   i =  1
cs:0017   891E0A00    mov   [000A],bx      sp  0000   d = 0
cs:001B   B44C        mov   ah,4C          ds  5424
cs:001D   CD21        int   21             ss 5434
ds:0000  CD 20  FF  9F  00  9A  F0  FE   -      □    Ü -   cs  5435
ds:0008  1D  F0  E0  01  45  1B  AA  01   ” -  Ó _ E _   _   ip  0000
ds:0010  45  1B 89  02  A0 15  12  07    E _ ë  _ á  § _ _   ss:0002   0000
ds:0018  01  01  01  00  02  FF  FF  FF   _ __  _            ss:0000 > A286
F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step   F9-Run F10-Menu
```

De asemenea, pe ecran apar doua linii de meniu:

- linia de meniu superioara, care cuprinde submeniurile:
  - System (=);
  - **File** (fisiere), care prin:
    - comanda **Open** permite *deschiderea unor fisiere executabile pentru depanare*,
    - comanda Quit (**[Alt] - X**) permite *iesirea din TD*;
  - View (vizualizare resurse);
  - Run (executie pas cu pas, pana la cursor, etc.);
  - Breakpoints (puncte de intrerupere a executiei programului);
  - Data (evaluare, urmarire);
  - Options (optiuni TD);
  - Window (modificare parametrii fereastra);
  - Help;
- linia de meniu inferioara, care cuprinde comenzile directe:
  - F1 - Help,
  - F2 - Bkpt (punct de intrerupere executie),
  - F3 - Mod,
  - **F4** - Here (*executie pana la cursor*),
  - F5 - Zoom (marire fereastra),
  - F6 - Next,
  - **F7** - Trace (*executie instructiune cu instructiune*),
  - F8 - Step (executie instructiune cu instructiune sarind peste proceduri),
  - F9 - Run (executie a programului),
  - F10 - Menu.

Apasand tasta **[Ctrl]** se poate observa ca meniul inferior se modifica. *Pentru a se specifica locatia de memorie care se doreste a fi afisata* se foloseste comanda **[Ctrl] G** urmata de adresa in formatul *segment : offset.*

Daca se doreste *afisarea segmentului de date incepand cu primul octet* se utilizeaza secventa **[Ctrl] G** urmata de **DS : 0**. Trecerea de la o subfereastra la alta se face apasand tasta **[Tab].**

5. Desfăşurarea lucrării

1. Se studiaza arhitectura microprocesorului Intel 8086.

2. Se porneste calculatorul si se intra in contul LAPSTn (n este numarul calculatorului). Se lanseaza in executie depanatorul simbolic TD cu comanda:

TD

3. Se studiaza comenzile sale si se experimenteaza utilizarea lor.

6. Teme si exercitii

1. Sa se calculeze adresele fizice corespunzatoare urmatoarelor adrese logice:
   a) 1205H : 709H,
   b) ABCDH : 89ABH,
   c) FFF0H : 0FFH,
   d) 3333H : 4444H,
   e) 8000H : 8000H.

2. Sa se calculeze componentele offset corespunzatoare urmatoarelor adrese fizice (se cunoaste componenta segment: 2000H):
   a) 20002H,
   b) 20010H,
   c) 20300H,
   d) 24000H,
   e) 2FFFFH.

3. Sa se calculeze componentele segment corespunzatoare urmatoarelor adrese fizice (se cunoaste componenta offset: 400H):
   a) 10400H,
   b) B0400H,
   c) 30800H,
   d) CDE00H,
   e) FFFF0H.

4. Care dintre urmatoarele adrese fizice apartin segmentului care are componenta segment 2400H:
   a) 33FFFH,
   b) 23000H,
   c) 27890H,
   d) 33000H,
   e) 34000H.