UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI

Instrumente Structurale
2007-2013

Platformă de e-learning și curriculă e-content pentru învățământul superior tehnic

Programare în limbaj de asamblare

17. Limbajul de asamblare

(definire constante, propoziții şi instrucțiuni).

Limbajul de asamblare

Limbajul de asamblare se caracterizează prin:

- utilizarea simbolurilor şi expresiilor simbolice pentru exprimarea codurilor operațiilor, a adreselor şi a operanzilor;
- utilizarea unor comenzi simbolice pentru controlul procesului de asamblare şi gestiunea resurselor calculatorului.

Elementele constitutive ale limbajului de asamblare sunt:

- alfabetul;
- cuvintele formate cu acest alfabet;
- propoziții formate din aceste cuvinte şi alfabet.

Alfabetul constă din:

- caracterele alfabetice: litere mici + litere mari;
- caractere numerice: 0 ÷ 9;
- caractere speciale: + – * / () [],. : ; @ $ _ ? ! etc.

Cuvintele sunt formate dintr-o succesiune de caractere din alfabet, construite după anumite reguli şi care au o anumită semnificație. Se împart în două categorii: identificatori şi constante.

Identificatorii sunt secvențe de lungime 1 ÷ 31 caractere alfanumerice şi speciale: ? _ @ $. Nu este permisă introducerea spațiilor libere într-un identificator. Un identificator trebuie să înceapă cu o literă sau cu unul din caracterele speciale. Dacă identificatorul începe cu punct (.), atunci are semnificația de comandă adresată asamblorului. Asamblorul nu deosebeşte literele mari de cele mici. Identificatorii sunt şi ei de două tipuri: standard sau predefiniți şi definiți de utilizator.

Identificatorii standard, sau cuvinte cheie, cum mai sunt denumiți, se utilizează pentru denumirea unor funcții şi argumente stabilite la definirea limbajului şi fac parte din vocabularul limbajului. Astfel de identificatori sunt:

- nume de instrucțiuni: MOV, ADD, SUB, INT etc.;
- nume de resurse: AX, BX,..., CS, DS,..., AH, AL etc.;
- nume de operatori: MOD, OFFSET, SEG, PTR, TYPE etc.;
- pseudoinstrucțiuni pentru asamblor: ASSUME, MODEL etc.

Identificatorii definiți de utilizator vor fi folosiți pentru:

- nume de variabile, adrese de instrucțiuni, operanzi;
- nume de proceduri, segmente, macroinstrucțiuni.

Definirea constantelor

Constantele pot fi numerice sau şiruri de caractere formate pe baza anumitor reguli şi cărora li se asociază o valoare. Constantele numerice pot fi întregi, reale sau zecimale (codificate binar).

Constantele întregi reprezintă numere întregi ce pot fi utilizate în funcție de context pentru date, adrese sau operanzi imediați. Constantele pot fi reprezentate în binar, octal, zecimal sau hexazecimal; reprezentarea într-una din aceste baze se specifică printr-una din literele B, Q, D sau H, care urmează după constantă:

- binar: 01101010B;
- octal: 152Q;
- zecimal: 106D sau 106;
- hexazecimal: 6aH.

Pentru a evita confuzia între constantele hexazecimale şi identificatori, întotdeauna o constantă hexazecimală trebuie să înceapă cu o cifră (de ex.: 0abcdH, pentru a nu fi confundată cu identificatorul abcdH).

Constantele zecimale pot fi urmate sau nu de litera D, deoarece se consideră implicit baza 10. Există o directivă de asamblare care permite stabilirea reprezentării într-o bază implicită a constantelor:

.RADIX < expresie >

unde < expresie > trebuie să aibă o valoare întreagă, în speță 2, 8, 10 sau 16, şi este evaluată în zecimal. Iată, de exemplu, valorile care se încarcă în registrul AX după utilizarea acestei directive (am folosit constante pentru expresia bazei):

```
.RADIX 2
mov         AX,100          ; (AX) = 0004H
.RADIX 8
mov         AX,100          ; (AX) = 0040H
.RADIX 10
mov         AX,100          ; (AX) = 0064H
.RADIX 16
mov         AX,100          ; (AX) = 0100H
```

Constantele întregi pot fi reprezentate în memorie pe 8, 16, 32 sau 64 biți.

Constantele reale sunt reprezentate în virgulă mobilă, prin mantisă şi exponent, în general sub forma:

[±] intreg. fractie [E ± exponent]

Asamblorul codifică aceste constante reale în reprezentarea cu virgulă mobilă pe 32, 64 sau 80 de biți, după cum se specifică tipul constantei:

- dd (define double word), 32 de biți, real simplă precizie;
- dq (define quadruple word), 64 de biți, real dublă precizie;
- dt (define ten bytes), 80 de biți, real temporar.

Codificarea în virgulă mobilă diferă de la un asamblor la altul, de la un compilator la altul, dar formatul cel mai răspândit este (IEEE):

S_m	exponent	mantisă

- S_m – semnul mantisei (1 bit);
- exponent – este exponentul numărului deplasat cu o anumită valoare, în funcție de tipul reprezentării:
  - 7fH, pentru formatul pe 32 de biți;
  - 3ffH, pentru formatul pe 64 de biți;
  - 3fffH, pentru formatul pe 80 de biți.
- exponentul pentru cele trei formate se reprezintă pe 8, 11 şi respectiv 15 biți;
- mantisa este normalizată, adică :
  - 1.00...0 <= mantisa <=1.11...1

şi se reprezintă, pentru cele trei formate, pe 23, 52 şi respectiv 64 de biți; de remarcat că pentru primele două formate nu se reprezintă prima cifră, deci prima unitate din fața virgulei, care este presupusă implicit, cu ajustarea corespunzătoare a exponentului. Pentru cel de-al treilea format se reprezintă şi prima unitate, cea din fața virgulei (punctului zecimal).

Exponentul, la reprezentarea internă, este deplasat, pentru a simplifica operațiile de comparare cu numere reale; în acest mod, procesorul lucrează numai cu valori pozitive pentru exponent. De exemplu, pentru valori reale simplă precizie (32 de biți), exponentul este în intervalul [1, 254], în loc de [-127, +126], cât ar fi fost domeniul exponentului pentru o reprezentare cu semn. Exponentul minim (0) este utilizat doar pentru reprezentarea valorii 0.0 şi pentru valori nenormalizate, iar cel maxim (255) este utilizat pentru a reprezenta o condiție de eroare (NaN – Not a Number), cu alte cuvinte codificarea cu exponent 11…11 nu reprezintă o valoare numerică.

Numerele negative diferă de numerele pozitive numai prin bitul de semn al mantisei.

Să considerăm, de exemplu, reprezentarea câtorva numere; am considerat, pentru simplitatea reprezentării valori care se reprezintă exact.

Pentru o declarație de forma

a) dd 1.0 ; 2^0
deci

$S_m = 0$;
exponent = 0 + 7f (deplasarea);
mantisa = 1.00...0;

se obține următoarea reprezentare internă:

S_m, exp, mant = 0, 01111111, 00...0 = 3F 80 00 00 H

b) dd -1.0 ; -2^0
deci

S_m, exp, mant = 1, 01111111, 00...0 = BF 80 00 00 H

c) dd 4.0 ; 2^2

deci

exponent = 2 + 7f = 81H;
S_m, exp, mant = 0, 10000001, 00...0 = 40 80 00 00 H

d) dd -0.625 ; $2^{-1} + 2^{-3}$;
deci

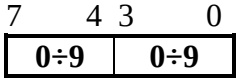
exponent = -1 + 7f = 7e;
mantisa = 1.0100...0;
S_m, exp, mant = 1, 01111110, 010...0 = BF 20 00 00 H

Valorile se memorează în ordinea inversă scrierii lor, adică la adresa cea mai mică octetul cel mai puțin semnificativ, iar la adresa cea mai mare (ultima) se memorează octetul cel mai semnificativ.

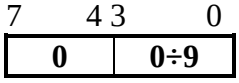
Un alt tip de constante îl reprezintă constantele zecimale codificate binar (BCD); procesorul poate opera cu două tipuri şi anume:

- zecimal împachetat;
- zecimal neîmpachetat (denumit şi format ASCII).

Primul format se caracterizează prin faptul că fiecare din cele două tetrade ale unui octet conține o cifră zecimală, deci valoarea unui astfel de octet este cuprinsă în intervalul 0 ÷ 99; cel de-al doilea format conține numai o singură cifră, pe tetrada mai puțin semnificativă, cealaltă fiind 0, ca în reprezentarea următoare:



a) BCD împachetat



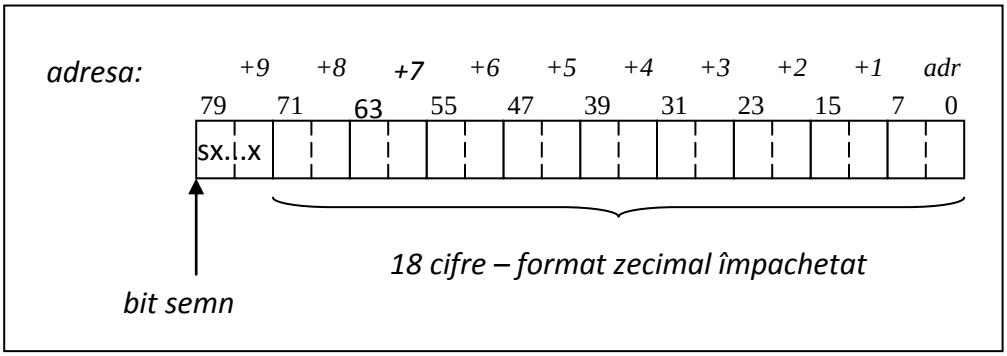
b) BCD neîmpachetat

În limbaj de asamblare, aceste constante sunt reprezentate în formatul împachetat, pe 80 biți, utilizând declarația de tip:

dt 12345678

pentru care se pot introduce maxim 18 cifre, deoarece octetul cel mai semnificativ conține, pe primul bit, semnul numărului, restul acestui octet (7 biți) fiind nefolosit. Memorarea se realizează după aceeaşi regulă, adică începând cu octetul cel mai puțin semnificativ şi terminând cu octetul cel mai semnificativ.

Formatul zecimal neîmpachetat mai este denumit şi format ASCII deoarece există o corespondență directă între formatul BCD neîmpachetat şi cel ASCII (singura diferență între acest format şi BCD este că prima tetradă conține valoarea 3, bineînțeles, pentru cifre).



Memorarea acestui număr se realizează conform regulii cunoscute, începând cu octetul cel mai puțin semnificativ la adresa *adr* şi terminând cu octetul cel mai semnificativ la adresa *adr + 9*.

De exemplu, pentru o declarație de forma: *dt 12345678*
reprezentarea internă va fi: *00 00 00 00 00 00 12 34 56 78*.

Constantele exprimate prin şiruri de caractere constau dintr-o succesiune de caractere incluse între apostrofuri (‘Exemplu’) sau între ghilimele ("sir caractere"). Dacă în interiorul unui şir de caractere un subşir trebuie să apară între apostrofuri sau ghilimele, atunci aceste caractere se dublează:

‘1 Dec’’1918’

Propoziții în limbaj de asamblare

O propoziție este o succesiune de cuvinte şi caractere din alfabetul limbajului de asamblare, construită după anumite reguli sintactice, de lungime maximă egală cu 128 de caractere şi care poate fi:

- instrucțiune;
- pseudoinstrucțiune.

O instrucțiune reprezintă o propoziție care este tradusă în cod maşină.

O pseudoinstrucțiune este o propoziție care nu se traduce în cod maşină, ea reprezentând o directivă pentru coordonarea modului de lucru al asamblorului:

- definiri constante;
- rezervări zone de memorie;
- controlul procesului de asamblare.

Instrucțiuni în limbaj de asamblare

Sintaxa generală a unei instrucțiuni este:

[eticheta :] [mnemonica [operanzi]] [; < comentarii >]

unde

etichetă – este un câmp care conține un nume simbolic format din maximum 31 de caractere alfanumerice şi caractere speciale _, ?, @, $. Primul caracter este o literă sau un caracter special. Fiecare etichetă are asociată o valoare numerică reprezentând adresa relativă în cadrul segmentului (sau grupului de segmente) din care face parte;

mnemonica – reprezintă numele instrucțiunii sau pseudoinstrucțiunii;

operanzi – este un câmp a cărui existentă şi format depinde de tipul instrucțiunii. Dacă instrucțiunea are doi operanzi, aceştia se numesc destinație şi sursă, şi apar în instrucțiune în această ordine . Operanzii pot fi şi expresii care conțin registre, constante, identificatori, şi operatori aritmetici, de indexare, logici, de deplasare, relaționali, de tip sau de conversie.