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

56. Directive de asamblare condiționată.

Directive de asamblare condiționată

Asamblorul permite utilizatorului să realizeze asamblarea doar a unei anumite părți a liniilor care compun modulul sursă. În principiu, pot exista două motive care să necesite o asamblare condiționată:

- un program general trebuie „adaptat“ pentru o anumită aplicație, în funcție de parametrii actuali incluzându-se anumite module, sau dimensionându-se zone de memorie corespunzător aplicației; altfel, prin includerea tuturor modulelor sau prin dimensionarea la maxim a zonelor de memorie, indiferent de aplicație, se ajunge la dimensionări neeconomice;
- un alt motiv ar putea fi necesitatea includerii în modulul obiect doar a unora dintre instrucțiunile prezente în textul sursă, datorită unor diferențe între procesoarele aceleiaşi familii (286, 386, 486 etc.); dacă programul conține instrucțiuni pentru oricare dintre procesoare, pentru a fi portabil cu maximum de eficiență (utilizare de instrucțiuni specifice unui anumit tip de procesor) fără intervenții în textul sursă, aceasta se poate obține prin includerea acestor instrucțiuni în secțiuni controlate de aceste directive de asamblare condiționată.

```
IF      conditie        ; conditia este o expresie ce poate
        secventa 1      ; fi evaluata, in momentul asamblarii,
[ELSE                   ; la o constanta 'TRUE' sau 'FALSE'
        secventa 2]
ENDIF
```

Aceste directive pot fi imbricate.
În mod asemănător sunt definite directivele:

```
IFE     expresie        ; daca expresia este zero, se executa secventa 1
                        ; altfel se executa secventa 2
IFB     argument        ; daca argumentul este blanc (' ') se
                        ; executa secventa 1, altfel se executa secventa 2
IFNB    <argument>      ; adica argumentul <> blanc se executa secventa 1,
                        ; altfel, argument =' ', executa secventa 2.
IFDIF <arg1>,<arg2>     ; se compara cele doua argumente caracter cu car
                        ; (senzitiv la tipul literelor: mari si mici), si
                        ; daca sunt diferite executa secv. 1, altfel secv.2
IFDIFI <arg1>,<arg2>    ; este asemanatoare cu directiva anterioara,
                        ; cu deosebirea ca ignora tipul literelor.
IFIDN <arg1>,<arg2>     ; se compara cele doua argumente caracter cu car
                        ; (senzitiv la tipul literelor: mari si mici), si
                        ; daca sunt identice executa secv.1, altfel secv.2
IFIDNI <arg1>,<arg2>    ; este asemanatoare cu directiva anterioara,
                        ; cu deosebirea ca ignora tipul literelor.
IFDEF <argument>        ; daca simbolul prezent ca argument este
                        ; definit executa secv.1, altfel secv.2
IFNDEF <argument>       ; daca simbolul prezent ca argument nu
                        ; este definit executa secv.1.
```

Ultimele două directive se utilizează pentru a evita o nouă definire a unui simbol, care a fost definit anterior, sau pentru stabilirea variantei de program dorite, ce urmează să se asambleze.

Întrucât asambloarele implică un proces în două etape, se poate folosi următoarea directivă de asamblare condiționată pentru a specifica asamblorului realizarea anumitor operații numai pe durata primei etape:

IF1
    secventa; evaluata, numai la prima parcurgere, la asamblare
ENDIF

Asemănător se defineşte şi directiva IF2, care acționează asupra declarațiilor din blocul asociat numai la pasul al doilea al procesului de asamblare.

De exemplu, următoarea declarație:

```
IF1
        INCLUDE C:\MACRO.LIB            ; include intreaga biblioteca
ENDIF
        PURGE putere, scrie_val         ; sterge macroinstructiiunile nedorite
```

va permite accesul la biblioteca specificată (MACRO.LIB) numai pe durata primei etape a procesului de asamblare. Directiva anterioară va include toate macroinstrucțiunile conținute în biblioteca respectivă. Dacă însă se doreşte includerea numai a anumitor macroinstrucțiuni, atunci după directiva anterioară trebuie plasată directiva PURGE urmată de numele macroinstrucțiunilor din biblioteca respectivă a căror includere nu se doreşte.

Exemple de utilizare a acestor directive:

1) Macroinstrucțiune pentru tipărirea unui caracter transmis ca parametru, dar care poate fi apelată fără nici un parametru, pentru afişarea caracterului spațiu.

```
tip_car     macro  car          ; utilizare:
ifb     <car>                   tip_car     'x'     ; genereaza
        mov    dl, ' '                  mov    dl, 'x'
else                                    mov    ah, 2
        mov    dl, car                  int    21h
endif                           tip_car     ; va genera
        mov    ah, 2                    mov    dl, ' '
        int    21h                      mov    ah, 2
endm                                    int    21h
```

2) O macroinstrucțiune recursivă, care depune registre în stivă; depunerea va înceta când în lista registrelor apare caracterul spațiu.

```
push_reg    macro  r1, r2, r3   ; utilizare:
        ifnb   <r1>                     push_reg    ax, bx, cx
```

```
                push    r1                      push    ax
                push_reg        r2, r3          push    bx
        endif                                   push    cx
endm
```

3) O macroinstrucțiune ce generează diferite instrucțiuni în funcție de parametrul furnizat. Un utilizator poate să o apeleze cu parametrul B sau W, pentru a preciza dacă să genereze instrucțiunea MOVSB sau MOVSW (implicit se generează MOVSB, dacă nu se specifică parametri).

```
movt    macro tip
        ifdn <tip>,<B>                  ; utilizare:    ifdn B
                rep movsb               ; sau:          ifdn W
                exitm                   ; sau:          ifdn
        endif
        ifdn <tip>,<W>
                rep movsw
        else            rep movsb       ; implicit, daca nu are parametru
        endif
endm
```

4) Evitarea redefinirii unor identificatori sau definirea condiționată:

```
                                var2    equ     2
ifndef  var1                    .....
        var1    equ     1       ifdef   var2
endif                                   var3    equ     3
                                endif
```

5) În ultimul exemplu, codul este asamblat condiționat şi generat pentru două procesoare diferite.

```
procesor = 386  ; este inițializat cu 8086 numai pentru acest procesor
. . . .
if procesor eq 386
   shl ax, 4        ; modifică numai AX
else                ; trebuie să fie procesorul 8086
   push cx          ; salvează CX, astfel că şi această secvență
   mov cl, 4        ; va modifica numai CX, precum
   shl ax, cl       ; secvența anterioară
   pop cx           ; reface CX
endif
```

Directive pentru generarea condiționată a erorilor

În conjuncție cu directivele de asamblare condiționată, există un set de directive pentru generarea de mesaje de eroare, în aceleaşi condiţii ca primele. Asamblorul generează mesajul de eroare la întâlnirea unei astfel de directive, dacă este îndeplinită condiția respectivă. Ele sunt utilizate, cu predilecție, în cadrul MI.

.ERR - generează mesaj de eroare, de exemplu:

```
        if      var1 lt var2
                .err    ; va genera mesaj dacă: var1 < var2
        endif
```

.ERRE **expresie**
.ERRNZ **expresie**
.ERRB **<argument>**
.ERRNB **<argument**
.ERRDIF **<arg1>,<arg2>**
.ERRDIFI **<arg1>,<arg2>**