Załącznik do ćwiczenia w środowisku MASM32

„Przesyłanie danych i zarządzanie danymi”

1 Pętle

Najszybszym sposobem organizacji pętli jest skorzystanie z instrukcji loop. Odejmuje ona od rejestru ECX (licznika) wartość 1, następnie sprawdza czy rejestr ECX ma wartość 0, jeżeli nie wkyonywany jest skok do etykiety podanej jako parametr.

Programista asemblerowy musi zadbać o to aby wewnątrz pętli wartość ECX nie uległa zmianie. Dobrym nawykiem jest odkładanie wartości ECX na stos na początku pętli i pobranie tej wartości do licznika przed użyciem instrukcji LOOP (na końcu pętli)

```
mov ecx, liczbaWykonań
petla:
    push ecx
    ;INSTRUKCJE DO WYKONANIA W PĘTLI
    pop ecx
loop petla
```

2 Operacje łańcuchowe

W operacjach łańcuchowych wykorzystujemy rejestry indeksowe ESI (Source Index) i EDI (Destination Index), których wartość się automatycznie zwiększa (lub zmniejsza w zależności od flagi kierunku)

2.1 MOVSB, MOVSW, MOVSD

- Rozkazy MOVSB, MOVSW i MOVSD kopiują dane z pamięci, lokalizacja wskazywana przez ESI do pamięci o adresie wskazywanym przez EDI.

- ESI i EDI są automatycznie zwiększane/zmniejszane o:

- MOVSB zwiększa/zmniejsza o 1
- MOVSW zwiększa/zmniejsza o 2
- MOVSD zwiększa/zmniejsza o 4

```
.data
source DWORD 0FFFFFFFFh
target DWORD ?
.code
mov esi,OFFSET source
mov edi,OFFSET target
movsd
```

2.2 Flaga kierunku DF

☐ Flaga kierunku decyduje o zwiększaniu lub zmniejszaniu ESI i EDI.

- DF = clear (0): zwiększa ESI i EDI
- DF = set (1): zmniejsza ESI i EDI

☐ Flaga kierunku może być zmieniana bezpośrednio przez rozkazy CLD i STD:

```
CLD ; clear Direction flag
STD ; set Direction flag
```

2.3 Prefiks REP

☐ Rep (a repeat prefix) może być wstawiony przed MOVSB, MOVSW or MOVSD
☐ ECX określa liczbę powtórzeń
☐ Przykład: Skopiuj 20 podwójnych słów ze źródła do celu

```
.data
source DWORD 20 DUP(?)
target DWORD 20 DUP(?)
.code
cld     ; direction = forward
mov ecx,LENGTHOF source ; set REP counter
mov esi,OFFSET source
mov edi,OFFSET target
rep  movsd
```

Użyj MOVSD do wykasowania pierwszego elementu tablicy. Wszystkie kolejne elementy przesuń na pozycję wcześniejszą:

array DWORD 1,1,2,3,4,5,6,7,8,9,10

```
.data
array DWORD 1,1,2,3,4,5,6,7,8,9,10
.code
cld
mov ecx,(LENGTHOF array) - 1
mov esi,OFFSET array+4
mov edi,OFFSET array
rep movsd
```

2.4 CMPSB, CMPSW, CMPSD

- CMPSB, CMPSW i CMPSD porównują operand w pamięci wskazywany przez ESI z operandem wskazywanym przez EDI.
  - CMPSB porównuje bajty
  - CMPSW porównuje słowa
  - CMPSD porównuje podwójne słowa
- Często używane prefiksy
  - REPE (REPZ) Rep if Equal
  - REPNE (REPNZ) Rep if Not Equal

Prefiksu REPE (repeat while equal) używamy np. do porównania dwóch elementów tablicy.

```
.data
source DWORD COUNT DUP(?)
target DWORD COUNT DUP(?)
.code
mov ecx,COUNT ; repetition count
mov esi,OFFSET source
mov edi,OFFSET target
cld ; direction = forward
 repe cmpsd ; repeat while equal
```

- SCASB, SCASW, SCASD porównuje wartości w AL/AX/EAX z odpowiednio bajtem, słowem, lub podwójnym słowem wskazywanym przez EDI.
- Użyteczne w wyszukiwaniu:
  - Określonego elementu w napisie lub tablicy
  - Pierwszego elementu nie pasującego do danego wzorca.

2.5 STOSB, STOSW, STOSD

- STOSB, STOSW, STOSD zapisują wartość z odpowiednio AL/AX/EAX, do miejsca w pamięci wskazywanego przez EDI.

- Przykład: wypełnienie tablicy wartością 0FFh

```
.data
Count = 100
string1 BYTE Count DUP(?)
.code
mov al,0FFh  ; value to be stored
mov edi,OFFSET string1    ; ES:DI points to target
mov ecx,Count     ; character count
cld      ; direction = forward
rep stosb     ; fill with contents of AL
```

2.6 LODSB, LODSW, and LODSD

- LODSB, LODSW,LODSD ładują wartość z pamięci o adresie wskazywanym przez ESI do odpowiednio AL/AX/EAX.

```
.data
array BYTE 1,2,3,4,5,6,7,8,9
.code
    mov esi,OFFSET array
    mov ecx,LENGTHOF array
    cld
L1: lodsb     ; „załaduj” bajt do AL
```

- REAL4
  - 4-byte IEEE short real
- REAL8
  - 8-byte IEEE long real
- REAL10
  - 10-byte IEEE extended real

Inny przykład - mnożenie

```
.data
array dword 1,2,3,4,5,6,7,8,9,1
multiplier DWORD 10
 .code
    cld      ; direction = up
    mov esi,OFFSET array          ; source index
    mov edi,esi      ; destination index
    mov ecx,LENGTHOF array        ; loop counter
L1: lodsd                      ; copy [ESI] into EAX
    mul multiplier        ; multiply by a value
    stosd                         ; store EAX at [EDI]
    loop L1
```

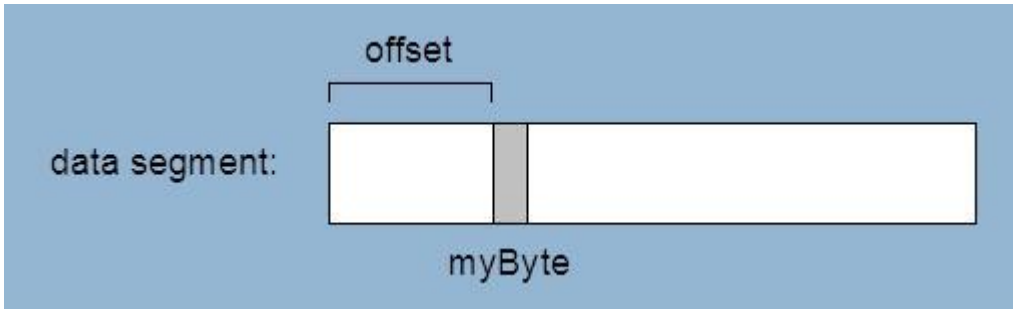
3 Typy danych

- BYTE, SBYTE (dawniej db)
  - 8-bit unsigned integer; 8-bit signed integer
- WORD, SWORD (dawniej dw)
  - 16-bit unsigned & signed integer

- DWORD, SDWORD (dawniej dd)
    - 32-bit unsigned & signed integer
- QWORD (dawniej dq)
    - 64-bit integer
- TBYTE (dawniej dt)
    - 80-bit integer
- REAL4
    - 4-byte IEEE short real
- REAL8
    - 8-byte IEEE long real
- REAL10
    - 10-byte IEEE extended real

4 Operator OFFSET

- OFFSET zwraca liczbę bajtów od początku segmentu do danej etykiety
    - Protected mode: 32 bits
    - Real mode: 16 bits



Przykład: Załóżmy, że segment danych zaczyna się w 00404000h

```
.data
bVal BYTE ?
wVal WORD ?
dVal DWORD ?
dVal2 DWORD ?
.code
mov  esi,OFFSET  bVal  ;  ESI  =  00404000
mov  esi,OFFSET  wVal  ;  ESI  =  00404001
mov  esi,OFFSET  dVal  ;  ESI  =  00404003
mov esi, OFFSET  dVal2 ; ESI = 00404007
```

5 Operator PTR

- Nadpisuje domyślny typ etykiety (zmiennej). Zapewnia elastyczność w dotarciu do części danej
- Little endian – pamiętamy o tej zasadzie

```
.data
myDouble DWORD 12345678h
.code
mov ax,myDouble              ; error – why?
mov ax,WORD PTR myDouble            ; loads 5678h
mov WORD PTR myDouble,4321h         ; saves 4321h
```

- PTR może być również użyty do przesłania do elementu o większy rozmiarze. CPU automatycznie odwróci kolejność.

```
.data
myBytes BYTE 12h,34h,56h,78h
.code
mov ax,WORD PTR [myBytes]     ; AX = 3412h
mov ax,WORD PTR [myBytes+2]       ; AX = 7856h
mov eax,DWORD PTR myBytes     ; EAX = 78563412h
```

6 lstrlenA

Funkcja lstrlenA zwraca liczbę znaków w buforze (ciąg znaków zakończony 0), wynik zwraca do rejstru EAX.

```
int WINAPI lstrlen(

_In_   LPCTSTR lpStr ing
);
```

LPCTSTR

The null-terminated string to be checked.

http://msdn.microsoft.com/en-us/library/windows/desktop/ms647492%28v=vs.85%29.aspx

7 Etykiety anonimowe

- Aby zwolnić programistę od wymyślenia dużej liczby etykiet lokalnych w języku MASM jest wprowadzona tzw. *etykieta anonimowa*.
- Etykieta anonimowa zawiera dwa znaki „@" i dwukropek. Dyrektywa @F wywołuje skok do przodu, do najbliższej etykiety anonimowej, a dyrektywa @B – skok do tyłu, do najbliższej etykiety anonimowej. Przykład stosowania etykiety anonimowej:

```
.CODE
...
 jnae @F ; skok do przodu
...
@@:
...
 jb @B ; skok do tyłu
```