Ćwiczenie

„Przesyłanie tablic”

Tematy ćwiczenia

- operacje na stosie,
- przesyłanie tablic

Sprawozdanie

Na każdym ćwiczeniu sporządza się sprawozdanie na bazie materiałów ćwiczenia. Bazowa zawartość sprawozdania musi być przygotowana w domu przed ćwiczeniem (sprawozdanie do ćwiczenia pierwszego jest przygotowywane w czasie ćwiczenia). W czasie ćwiczenia do sprawozdania są dodawane wyniki testowania.

Treść sprawozdania:

strona tytułowa,

spis treści sporządzony za pomocą *Word'*a,

dla każdego punktu rozdziały "Zadanie ", "Opracowanie zadania" (rozdział z tekstem programu i komentarzami), "Testowanie" (rozdział z opisem danych wejściowych i wynikami testowania, w tym zrzuty aktywnego okna).

Nazwa (bez polskich liter, żeby można było archiwizować) pliku ze sprawozdaniem musi zawierać nazwę przedmiotu jako "PN", grupę, numer ćwiczenia i nazwisko studenta.

Pliki ze sprawozdaniem są przekazywane do archiwum grupy.

a) Przesyłanie przez stos

Zadanie

Napisać program, w którym wykonać działania następujące.

Zdefiniować i wyświetlić bajtową tablicę tekstową z tekstem - funkcją `...`[1], który zawiera 7 znaków plus znak '\0' (koniec wierszu).

Zdefiniować buforową bajtową tablicę tekstową.

Przestawić miejscami połówki zawartości tablicy bazowej metodą zapisywania na stos i zdejmowania ze stosu. Wynik przypisać do tablicy buforowej.

Wyświetlić zawartość tablicy buforowej.

Opracowanie zadania

<<tekst programu>>

Testowanie

<<zrzut ekranu MS DOS>>

---

b) Przesyłanie tablic

Zadanie

Napisać program, w którym wykonać działania następujące.

Zdefiniować i wyświetlić bajtową tablicę tekstową z tekstem - funkcją `...`[2], który zawiera 7 znaków plus znak '\0' (koniec wierszu).

Zdefiniować buforową bajtową tablicę tekstową.

Przypisać tekst z tablicy bazowej do tablicy buforowej stosując rozkaz „`rep movs`”.

---

[1] `z tabeli wariantów`

[2] `z tabeli wariantów`

Wyświetlić zawartość tablicy buforowej.

Opracowanie zadania

<<tekst programu z komentarzami>>

Testowanie

<<zrzut ekranu MS DOS>>

Tabela wariantów

Np	Funkcja		Np	Funkcja		Np	Funkcja		Np	Funkcja
1	A + B + C + D		**9**	A + B * C + D		**17**	A + B + C – D		**25**	A + B * C - D
2	A - B + C + D		**10**	A - B * C + D		**18**	A - B + C - D		**26**	A - B * C - D
3	A * B + C + D		**11**	A * B * C + D		**19**	A * B + C - D		**27**	A * B * C – D
4	A / B + C + D		**12**	A / B * C + D		**20**	A / B + C - D		**28**	A / B * C - D
5	A + B - C + D		**13**	A + B / C + D		**21**	A + B - C - D		**29**	A + B / C - D
6	A - B - C + D		**14**	A - B / C + D		**22**	A - B - C - D		**30**	A - B / C - D
7	A * B - C + D		**15**	A * B / C + D		**23**	A * B - C - D		**31**	A * B / C - D
8	A / B - C + D		**16**	A / B / C + D		**24**	A / B - C - D		**32**	A / B / C - D

Wskazówki

Adresowanie operandów

W 32-bitowych procesorach Intel operand operacji może znajdować się w kodzie instrukcji, w rejestrze lub w komórce pamięci.

Jeżeli operand znajduje się w kodzie instrukcji, to mówimy o **adresowaniu natychmiastowym**. Przy **adresowaniu domyślnym** rozkaz nie zawiera adresu i operandu.

Jeżeli rozkaz zawiera numer (indeks) rejestru, a w rejestrze znajduje się operand, to nazywamy takie adresowanie **„bezpośrednim rejestrowym”**. Jeżeli procesor interpretuje zawartość rejestru jako adres, to mamy do czynienie z **adresowaniem rejestrowym pośrednim**.

Rozkaz z adresem danej w pamięci powoduje wykorzystanie **adresowania bezpośredniego**.

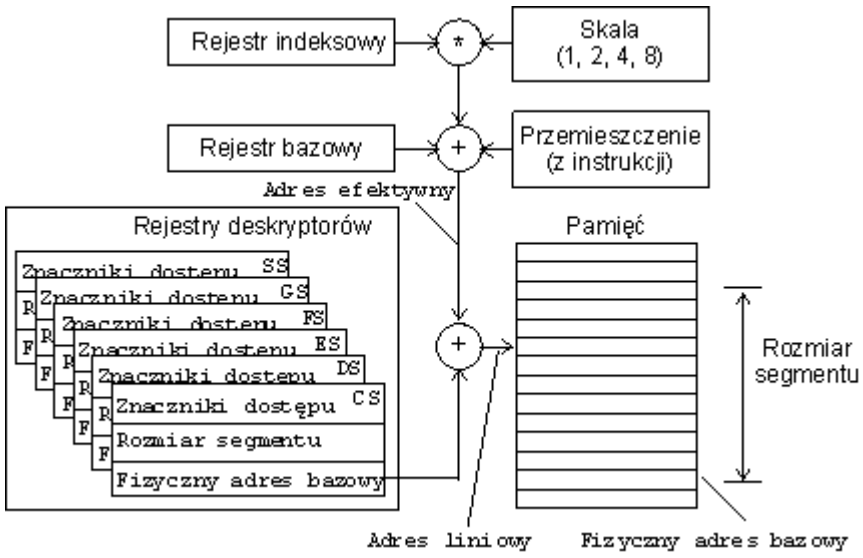
Adresowaniem bazowym nazywamy tryb, kiedy procesor oblicza adres operandu korzystając z zawartości jednego z rejestrów bazowych EBP, ESP.

W przypadku obliczenia adresu z wykorzystaniem zawartości rejestru indeksowego DI lub SI ma miejsce tryb **adresowania indeksowego**.

Jeżeli przy obliczeniu adresu procesor korzysta z zawartości rejestrów bazowego i indeksowego, to nazywamy tryb **adresowania bazowym indeksowym**.

Jeżeli rozkaz zawiera przesunięcie i wskazuje na zastosowanie rejestrów bazowego i indeksowego, to adresowanie będzie **„bazowym indeksowym z przesunięciem”**.

Obliczenie w 32-bitowych procesorach adresu realnego (ang. *linear address*) ilustruje rys. 3.



Rys. 3. Kolejność obliczenia adresu realnego w procesorach 32-bitowych

Określenie typu wyrażenia

Do określenia typu wyrażenia jest używany operator PTR.

Składnia operatora PTR:

```
_Typ PTR _Wyrażenie
```

Przykład, w którym operator PTR jest użyty do operacji na znakach w tablicy tekstowej:

```
.DATA
tabl DWORD „abc”,0
.CODE
    mov AL, BYTE PTR tabl[0]
    mov BYTE PTR tabl[2], AL
```

Przesyłanie danych

Daną można przesyłać z rejestru do rejestru, z rejestru do komórki pamięci lub z komórki pamięci do rejestru. Nie jest możliwe bezpośrednie przesyłanie danej między dwoma komórkami pamięci.

Jedna instrukcja zawsze przesyła jedną daną, ale część instrukcji jest przystosowana do powtórzenia przesyłania.

Zmiana formatu danej w procesie przesyłania

Przesyłanie danej można połączyć ze zmianą formatu, a mianowicie z rozszerzeniem znaku (instrukcja movsx) lub z zerowaniem starszych bitów (instrukcja movzx).

Rozkaz „`movsx`” w procesie przesyłania danej rozszerza znak z bajta na słowo (podwójne słowo) lub ze słowa na podwójne słowo.

Podobny rozkaz „`movzx`” rozszerza daną zerami w stronę starszego bajta lub słowa.

Przesyłanie adresów i przesunięć segmentowych

Do operacji z adresami i przesunięciami segmentowymi służą specjalne instrukcje lea, lds, les, lfs, lgs, lss.

Wykonując rozkaz „`lea`” procesor przesyła do miejsca przeznaczenia przesunięcie adresu komórki pamięci.

Rozkazy procesora Intel

W dokumentacji firmy Intel rozkazy 32- bitowych procesorów są podzielone na grupy:

- `Data Transfer` – przesyłanie danych,
- `Segment Control` – sterowanie segmentami,
- `Flag Control` – sterowanie znacznikami,
- `Arithmetic` – operacje arytmetyczne,
- `Logic` – operacje logiczne (bitowe),
- `Shift/Rotate` – przesuwanie bitowe,
- `String Manipulation` – operacje z wierszami (tablicami),
- `Bit Manipulation` – operacje bitowe,

- `Control Transfer` – przejście sterowane,
- `Conditional Jumps` – skoki warunkowe,
- `Conditional Byte Set` – warunkowe ustawienie bajtu,
- `Interrupt Instructions` – rozkazy przerwań,
- `Processor Control` – sterowanie procesorem,
- `Prefix Bytes` (bajty prefiksu),
- `Protection Control` (sterowanie ochroną),
- `High Level Language Support` – potrzymanie języka wysokiego poziomu,
- `Operating System Support` – potrzymanie systemu operacyjnego,
- `Processor Extension Instruction` (instrukcje koprocesora),
- `MMX Unit Instructions` (instrukcje jednostki MMX).

Kodowanie rozkazów procesorów Intel jest przedstawione w tabelach przytoczonych niżej, gdzie oznaczono:

`A` – rejestr-akumulator EAX (AX, AH, AL),

`EA` – adres efektywny,

`Num` – numer portu,

`param8/16/32` – parametr 8-, 16- lub 32-bitowy,

`przes8/16/32` – przesunięcie 8-, 16- lub 32-bitowe,

`P` – pamięć,

`R` – rejestr,

`r8/16/32` – rejestr 8-, 16- lub 32-bitowy,

`R/P` – rejestr lub pamięć,

`Rs` – rejestr segmentowy.

Grupa `Data Transfer` (przesyłanie danych)

Tabela 7

Kodowanie rozkazów grupy `Data Transfer` (przesyłanie danych)

Mnemonik i operandy	Bajt 0	Bajt 1	Bajt 2

mov R/P, R
mov R, R/P
mov R/P, Rs
mov Rs, R/P
mov A, P
mov P, A
mov R, dana
mov R/P,dana
	1000100w
	1000101w
	10001100
	10001110
	1010000w
	1010001w
	1011wreg
	1100011w
		mod_reg_r/m
		mod_reg_r/m
		mod_sr3_r/m
		mod_sr3_r/m
		przes
		przes
		dana
		mod_000_r/m
			
			dana

movsx R, R/P	00001111	1011111w	mod_reg_r/m
movzx R, R/P	00001111	1011011w	mod_reg_r/m
push R/P push R push Rs push FS/GS push dana	11111111 01010reg 000s2110 00001111 011010s0	mod_110_r/m 10_sr3_000 dana	
pop R/P pop R pop Rs pop FS/GS	10001111 01011reg 000s2111 00001111	mod_000_r/m 10_sr3_001	
pusha; pushad	01100000		
popa; popad	01100001		
xchg R/P, R; xchg R, R/P xchg R, A; xchg A, R	1000011w 10010reg	mod_reg_r/m	
in A, Num in A, DX	1110010w 1110110w	Num	
out Num, A out DX, A	1110011w 1110111w	Num	
lea R, P	10001101	mod_reg_r/m	

Wykonując rozkaz „`mov odbiorca, źródło`” (tab. 7) procesor może przesłać daną z rejestru do rejestru, z komórki pamięci do rejestru, w tym do akumulatora, i odwrotnie, oraz może zapisać do rejestru lub do komórki pamięci daną bezpośrednią. Dana może być 8-, 16- lub 32 bitowa. Nie istnieje możliwość przesyłania danych bezpośrednio między komórkami pamięci. W przypadku przesyłania danej do rejestru segmentowego ma miejsce zakaz przerwań.

Przykłady:

```
mov EAX,0
mov ECX,10
mov EBX, OFFSET zmienna
```

Zamianę miejscami zawartości rejestru i drugiego rejestru, lub rejestru i akumulatora, lub rejestru i komórki pamięci wykonuje rozkaz „`xchg`”.

Rozkaz „`movsx`” w procesie przesyłania danej rozszerza znak z bajta na słowo (podwójne słowo) lub ze słowa na podwójne słowo.

Podobny rozkaz „`movzx`” rozszerza daną zerami w stronę starszego bajta lub słowa.

Wykonując rozkaz „`lea`” procesor przesyła do miejsca przeznaczenia przesunięcie adresu komórki pamięci.

W instrukcjach przesyłania danych można wskazywać rejestr segmentowy przed nazwą zmiennej. Na przykład przy kompilacji instrukcji:

```
mov EAX, ES:symb
```

asembler wykorzysta zawartość rejestru segmentowego ES.

Przesyłanie danej na stos / ze stosu

Przesyłanie danej na stos i ze stosu przy wykonaniu instrukcji grupy Data Transfer i grupy Flag Control jest równoważne przesyłaniu danej między rejestrem i komórką pamięci, ponieważ stos programowy to grupa komórek pamięci.

Przesyłanie danych przez stos często jest używane do przesyłania danej między komórkami pamięci, na przykład:

```
.CODE
 push zm1
 pop temp ;do zmiennej "temp" jest przypisana zmienna "zm1"
```

W procesie wykonania rozkazów „push" i „pop" procesor ładuje daną na stos lub zdejmuje daną ze stosu.

Wierzch stosu znajduje się pod adresem SS:ESP, a stos rośnie w stronę mniejszych adresów.

Wykonując rozkaz „push" procesor zmniejsza ESP o dwa lub cztery w zależności od typu procesora i zapisuje operand na wierzch stosu.

W przypadku odkładania na stos bajta ma miejsce rozszerzenie znaku.

Rozkaz „pop" powoduje odczyt danej z wierzchu stosu spod adresu SS:ESP, a następnie zwiększenie ESP o dwa lub cztery w zależności od typu procesora. Operacja „pop" nie jest możliwa, jeżeli miejscem przeznaczenia danej jest rejestr segmentowy CS.

Rozkazy „pusha/pushad" i „popa/popad" są bardzo przydatne na początku i w końcu podprogramu, ponieważ ładują na stos i zdejmują ze stosu grupę rejestrów ogólnego przeznaczenia. Kolejność odkładania na stos zawartości rejestrów:

dla pusha: AX, CX, DX, BX, SP (przed odkładaniem), BP, SI, DI;

dla pushad: EAX, ECX, EDX, EBX, ESP (przed odkładaniem), EBP, ESI, EDI.

Oczywiście, że kolejność zdejmowania ze stosu zawartości rejestrów jest odwrotna:

dla popa: DI, SI, BP, SP, BX, DX, CX, AX;

dla popad: EDI, ESI, EBP, ESP, EBX, EDX, ECX, EAX.

Przesyłanie danej do portu / z portu

Do wprowadzenia danej z portu jest używany rozkaz „in", a do wyprowadzenia danej do portu - rozkaz „out". Jeżeli numer portu znajduje się w granicach 0 – 255, to stosuje się rozkaz z bezpośrednim adresowaniem operandu. W przypadku, gdy numer portu jest większy niż 255, stosuje się rejestr DX i adresowanie pośrednie rejestrowe. Maksymalnie możliwy numer portu w komputerach PC jest równy 1023.

Pobieranie danej z portu i ładowanie danej do portu przy wykonaniu instrukcji in i out posiada ograniczenie: przesyłanie jest możliwe tylko między rejestrem – akumulatorem a portem.

Grupa `String Manipulation` (operacje z wierszami, tablicami)

Tabela

Kodowanie rozkazów grupy `String Manipulation` (operacje z wierszami, tablicami)

Mnemonik i operandy	Bajt 0	Bajt 1
`cmps P1, P2`	`1010011w`	
`scas P`	`1010111w`	
`repe cmps P1, P2`	`11110011`	`1010011w`
`repe scas P`	`11110011`	`1010111w`
`repne cmps P1, P2`	`11110010`	`1010011w`
`repne scas P`	`11110010`	`1010111w`
`ins P1, DX`	`0110110w`	
`outs P1, DX`	`0110111w`	
`lods P`	`1010110w`	
`stos P`	`1010101w`	
`movs P1, P2`	`1010010w`	
`rep ins P1, DX`	`11110010`	`0110110w`
`rep outs P1, DX`	`11110010`	`0110111w`
`rep lods P`	`11110010`	`1010110w`
`rep stos P`	`11110010`	`1010101w`
`rep movs P1, P2`	`11110010`	`1010010w`
`xlat P`	`11010111`	

Rozkazy „`cmps`” i „`scas`” tej grupy są przystosowane do wielokrotnego porównania dwóch argumentów.

Wykonując rozkaz „`cmps`” procesor porównuje dwa operandy przez odejmowanie, w wyniku którego ma miejsce ustawienie znaczników: OF, SF, ZF, AF, PF, CF, - ale wynik odejmowania nie jest zapisywany. Operand pierwszy jest pobierany z komórki pamięci z pod adresu ES:EDI, a operand drugi - z komórki pamięci z pod adresu DS:ESI. W końcu operacji zawartość EDI i ESI zwiększą się lub zmniejszą się w zależności od znacznika kierunku DF w rejestrze znaczników (DF=0 - zwiększenie, DF=1 - zmniejszenie) o 1 dla operacji bajtowej, 2 dla operacji nad słowem, lub 4 dla operacji nad podwójnym słowem.

Przy wykonaniu rozkazu „`scas`” procesor też porównuje operandy, przy tym procesor operuje z domyślnym drugim operandem z rejestru-akumulatora. Operand pierwszy jest pobierany z komórki pamięci z pod adresu ES:EDI. Wynik odejmowania też nie jest zapisywany, ma miejsce tylko ustawienie znaczników: OF, SF, ZF, AF, PF, CF. W końcu operacji rejestr EDI zwiększa się lub zmniejsza się w zależności od znacznika kierunku DF w rejestrze znaczników (DF=0 - zwiększenie, DF=1 - zmniejszenie) o 1 dla operacji bajtowej, 2 dla operacji nad słowem, lub 4 dla operacji nad podwójnym słowem.

Rozkazy „`repe cmps`”, „`repne cmps`”, „`repe scas`” i „`repne scas`” zawierają przedrostki „`repe`” i „`repne`”.

Przedrostek „`repe`” (nazwa od ang. *repeat equal*) powoduje powtórzenie operacji porównania „`cmps`” lub „`scas`”. Ilość powtórzeń musi być zapisana do rejestru CX. Po każdemu porównaniu zawartość rejestru CX jest zmniejszana o 1. Cykl będzie zakończony przy zerowej zawartości rejestru CX. Drugim warunkiem zakończenia cyklu jest niezerowa wartość znacznika ZF.

Stosując rozkazy „`repe cmps`” i „`repe scas`” można porównywać elementy dwóch tablic lub element tablicy i zawartość rejestru-akumulatora. Ilość porównań jest ograniczona początkowej zawartością rejestru CX, i porównania mogą być zakończone wcześniej na pierwszych nierównych elementach dwóch tablic (dla „`repe cmps`”) lub elementu tablicy i zawartości rejestru-akumulatora (dla „`repe scas`”).

Podobny przedrostek „`repne`” (nazwa od ang. *repeat not equal*) też powoduje powtórzenie operacji porównania „`cmps`” lub „`scas`” w ilości równej początkowej zawartości rejestru CX. Dodatkowy warunek zakończenia porównań – zerowa wartość znacznika ZF. Innymi słowy porównania będą zakończone na pierwszych równych elementach dwóch tablic (dla „`repne cmps`”) lub elementu tablicy i zawartości rejestru-akumulatora (dla „`repne scas`”).

Do wprowadzenia elementu tablicy z portu i wyprowadzenia elementu tablicy do portu służą rozkazy „`ins`” i „`outs`” odpowiednio. Element tablicy jest pobierany z komórki lub zapisywany do komórki pamięci z pod adresu ES:EDI, a numer portu musi być zapisany do rejestru DX. W końcu operacji procesor zmienia zawartość

rejestru EDI w zależności od znacznika kierunku DF: zwiększa przy DF równym 0 lub zmniejsza przy DF równym 1.Wielkość zmiany zawartości rejestru indeksowego EDI zależy od rozmiaru danej i może być równa 1, 2 lub 4.

Wykonując rozkazy „`lods`" i „`stos`" procesor przesyła daną między rejestrem-akumulatorem i komórką pamięci. Adres komórki pamięci znajduje się w rejestrach:

- DS:ESI dla rozkazu „`lods`",
- ES:EDI dla rozkazy „`stos`".

Zawartość rejestru indeksowego ESI lub EDI jest zwiększana, jeśli znacznik kierunku DF ma wartość zerową, i zmniejszana, jeśli znacznik kierunku DF ma wartość niezerową. Zwiększenie lub zmniejszenie zawartości rejestru EDI jest równe 1, 2 lub 4 w zależności od rozmiaru danej.

Rozkaz „`movs`"

Rozkaz „`movs`" przesyła daną 8-, 16- lub 32-bitową między dwoma komórkami pamięci, adresy których muszą być zapisane do rejestrów DS:ESI (źródło danej) i ES:EDI (miejsce przeznaczenia).

Po zakończeniu przesyłania zawartość obydwóch rejestrów indeksowych ESI i EDI zmienia się w kierunku, który zależy od wartości znacznika kierunku DF: zwiększa się, jeśli znacznik kierunku DF ma wartość zerową, i zmniejsza się, jeśli znacznik kierunku DF ma wartość niezerową.

Przedrostek „`rep`" (nazwa od ang. *repeat*) w rozkazach „`rep ins`", „`rep outs`", „`rep lods`", „`rep stos`" i „`rep movs`" powoduje powtórzenie operacji. Procesor powtarza wykonanie rozkazu „`rep ins`", „`rep outs`", „`rep lods`", „`rep stos`" lub „`rep movs`" dopóki zawartość rejestru CX jest niezerowa. Po każdemu wykonaniu rozkazu zawartość rejestru CX jest zmniejszana o 1.

Rozkaz „`xlat`" (nazwa na pewnie od ang. ***ex**change **l**ow **a**ccumulator **t**able*) wykonuje zamianę zawartości rejestru AL na bajt z komórki z pod adresu [DS:EBX+AL]. Jak widać poprzednia zawartość rejestru AL stosuje się jako indeks bajtowy w tablicy z pod adresu DS:EBX.