Kamil Skarżyński

Programowanie niskopoziomowe
Sterowanie przebiegiem programu
Laboratorium 05

1. Wprowadzenie

Podczas laboratoriów zapoznamy się z:

1. Instrukcjami cmp i test
2. Rejestrem flagowym i znaczeniem flag,
3. Skokami warunkowymi,
4. Makrami,
5. Generatorem liczb pseudo losowych,
6. Sposobem dołączania bibliotek.

2. Zadania

1. Wczytaj kolejn 2 liczby podane przez użytkownika. Porównaj je i wyświetl odpowiedni komunikat (2 punkty):
   a) Podane liczby są równe,
   b) Podane liczby nie są równe.
2. Wczytaj kolejno 2 liczby podane przez użytkownika. Porównaj je i wyświetl odpowiedni komunikat (2 punkty):
   a) Pierwsza liczba jest większa,
   b) Liczby są równe,
   c) Pierwsza liczba jest mniejsza.
3. Wyświetl 10 liczb pseudolosowych.
4. Napisz mini grę. Program losuje liczbę z zakresu 0-X. Przy uruchomieniu użytkownik wprowadza liczbę X. Następnie użytkownik zgaduje wylosowaną liczbe i otrzymuje jeden z komunikatów (4 punkty):
   a) Trafiłeś za N razem,
   b) Nie trafiłeś. Podaj większą liczbę,
   c) Nie trafiłeś. Podaj mniejszą liczbę,
5. Porównaj 2 obiekty jak w zadaniu 2. Obiekty wybierz odpowiednio z tabeli wariantów (2 punkty).

3. Warianty

Tabela wariantów do zadani 5.

Tabela 1. Tabela flag

1	Pole prostokąta	2 parametry(dłuugość a, długość b)
2	Długość odcinka	4 parametry(pocz 1 koniec 1, pocz 2 koniec 2)
3	Pole rombu	2 parametry(przekątna a, przekątna b)
4	Długość imienia	2 parametry(imię 1, imię 2)
5	Długość nazwiska	2 parametry(nawzisko 1, nazwisko 2)

4. Pomoc

4.1. Instrukcje porównania

4.1.1. CMP

Instrukcja CMP porównuje operanty i ustawia odpowiednio flagi. Działa identycznie jak:

```
SUB destination, source
```

Z tą różnicą, że wynik nie jest nigdzie zapamiętywany(w przypadku sub jest w destination) ,a mimo to ustawiane są flagi. Składnia:

```
CMP r8/m8, r8/imm/m8 ;drugi operand moze byc
;miejscem w pamieci tylko gdy pierwszy nim jest
CMP r16/m16, r16/imm/m16 ;jak wyzej
CMP r32/m32, r32/imm/m32 ;jak wyzej
```

Przykład wykorzystania:

```
MOV EAX, 1
MOV EDX, 1
CMP EAX,EDX
```

4.1.2. TEST

Wykonuje operacje AND(koniunkcja) na odpowiadających sobie bitach operandów. Działa jak instrukcja AND, jednak nie zapisuje nigdzie wyniku operacji. Ustawia jedynie stosowne flagi jak robi to instrukcja AND:

```
TEST r8/m8, r8/imm/m8 ;drugi operand moze byc
;miejscem w pamieci tylko gdy pierwszy nim jest
TEST r16/m16, r16/imm/m16 ;jak wyzej
TEST r32/m32, r32/imm/m32 ;jak wyzej
```

Przykład wykorzystania:

```
MOV EAX, 1
MOV EDX, 1
TEST EAX,EDX
```

4.2. Rejestr flagowy i flagi

W środowisku VS mamy możliwość podejżenia bezpośrednio rejestru EFL, lub wybranych flag. By zobaczyć wybrane flagi klikamy prawym przyciskiem myszy na pustą przestrzeń gdzie już widzimy dostępne rejestry(w okienku Registers). A następnie wybieramy Flags.
Znaczenie flag:

Tabela 2. Tabela flag

Znaczenie	Skrót
Overflow	OV
Direction	UP
Interrupt	EI
Sign	PL
Zero	ZR
Auxiliary	AC
Parity	PE
Carry	CY

4.3. skoki warunkowe

Dostępne jest 30 mnemoników dla wygodny oraz czytelności kodu. Faktycznie istnieje ich jednak tylko 16.
Dostępne instrukcje skoków na podstawie flag:

Tabela 3. Tabela skoków warunkowych

Mnemonik	Warunek testowany	Opis
jo	OF = 1	overflow
jno	OF = 0	not overflow
jc, jb, jnae	CF = 1	carry / below / not above nor equal
jnc, jae, jnb	CF = 0	not carry / above or equal / not below
je, jz	ZF = 1	equal / zero
jne, jnz	ZF = 0	not equal / not zero
jbe, jna	CF or ZF = 1	below or equal / not above
ja, jnbe	CF or ZF = 0	above / not below or equal
js	SF = 1	sign
jns	SF = 0	not sign
jp, jpe	PF = 1	parity / parity even
jnp, jpo	PF = 0	not parity / parity odd
jl, jnge	SF xor OF = 1	less / not greater nor equal
jge, jnl	SF xor OF = 0	greater or equal / not less
jle, jng	(SF xor OF) or ZF = 1	less or equal / not greater
jg, jnle	(SF xor OF) or ZF = 0	greater / not less nor equal

4.4. Pętle

Do tej pory przy konstrukcji pętli wykorzystywaliśmy:

mov ECX, 100

```
label:
        ;kod petli
loop label
```

Instrukcja loop odejmuje od rejestru ECX 1 i sprawdza czy nie jest zerem, jeżeli nie jest to wykonuje skok do etykiety. Skok musi być skokiem krótkim(do 127 bajtów). Wydajniejszą konstrukcją jest kod poniżej z wykorzystaniem flag:

```
mov EDX,100

label:
        ;kod petli
        dec EDX ;gdy osiagnie 0 flaga zero zostanie ustawiona
jnz label
```

4.5. Makra

Poniżej przedstawione zostaną przykładowe makra z których można korzystać w kolejnych laboratoriach, jeśli teśc zadania nie wskazuje inaczej. Operatory wykorzystywane w wyrażeniach porównujących:

Tabela 4. Tabela operatorów porównań

expr1 == expr2	Zwraca true gdy expr1 jest równe expr2.
expr1 != expr2	Zwraca true gdy expr1 nie jest równe expr2.
expr1 <= expr2	Zwraca true gdy expr1 jest większe niż expr2.
expr1 >= expr2	Zwraca true gdy expr1 jest większe lub równe z expr2.
expr1 <expr2	Zwraca true gry expr1 jest mniejsze niz expr2.
expr1 <= expr2	Zwraca true gdy expr1 jest mniejsze lub równe z expr2.
!expr	Zwraca true gdy expr jest false.
expr1 && expr2	Przeprowadza logiczny AND pomiędzy expr1 i expr2.
expr1 || expr2	Przeprowadza logiczny OR pomiędzy expr1 i expr2.
expr1 & expr2	Przeprowadza bitowy AND pomiędzy expr1 i expr2.
CARRY?	Zwraca true gdy flaga przeniesienia jest ustawiona.
OVERFLOW?	Zwraca true gdy flaga przepełnienia jest ustawiona.
PARITY?	Zwraca true gdy flaga parzystości jest ustawiona.
SIGN?	Zwraca true gdy flaga znaku jest ustawiona.
ZERO?	Zwraca true gdy flaga zera jest ustawiona.

4.5.1. .IF przykłady

```
.IF variable == value
        ;kod
.ENDIF
```

```
.IF variable == value
        ;kod
.ELSEIF variable != another_value
```

```
        ;inny kod
.ELSE
        ;kod else
.ENDIF



.IF variable > 50 && variable < 100
        ;number in range
.ELSE
        ;number out of range
.ENDIF



.IF EAX > EBX && EAX > ECX
        ;kod
.ELSE
        ;kod
.ENDIF
```

4.5.2. .REPEAT przykłady

```
.REPEAT
        sub EAX, 1
.until EAX < 1
```

4.5.3. .WHILE przykłady

```
.WHILE EAX < 10
        inc EAX
.ENDW
```

4.6. ndrand - liczby pseudolosowe

W wygenerowania liczby pseudolosowej możemy wykorzystać procedurę *nrand*, która znajduje się w bibliotece masm32.lib i przyjmuje 1 parametr, zakres. Procedura ta generuje liczbę pseudolosową na podstawie ziarna oraz numeru wywołania od ostatniej inicjalizacji generatora. W celu inicjalizacji generatora wykorzystujemy procedurę *nseed* dla której parametrem jest wartość ziarna. By uzyskać "losową" wartość ziarna możemy użyć bezparametrowej procedury *GetTickCount* która zwraca czas w ns od ostatniego uruchomienia procesora. Przykład użycia:

```
GetTickCount PROTO
nseed PROTO :DWORD
nrandom PROTO :DWORD

.data
range                           DWORD 100
wylosowanaLiczba        DWORD 100
.code
```

```
call GetTickCount ;zwraca w czas w milisekundach
;od ostatniego uruchomienia systemu
push eax ; wrzucamy czas od uruchomienia na stos
call nseed ;ustawienie ziarna generatora liczb
push zakres
call nrandom ;zwraca w eax liczbe z zakresu 0-range
mov wylosowanaLiczba ,EAX
```

4.7. Dołączanie bibliotek

W celu dodania biblioteki zapewniającej losowanie liczb pseudolosowych, musimy pobrać ze strony bibliotekę: masm32.lib. Następnie klikamy w solucji prawym przyciskiem na projekt i klikamy properties. Następnie Configuration Properties→Linker→General→Additional Library Directories i wybieramy katalog w którym znajduje się nasz plik.
Dzięki temu mamy teraz możliwość wskazania linkerowi które biblioteki z tego folderu powinien wykorzystać, w tym celu: Przechodzimy do Configuration Properties→Linker →Input→Additional Dependencies klikamy edit I dopisujemy masm32.lib. Lub możemy dodać dyrektywę "includelib masm32.lib".