Ćwiczenie 1. Zmienne, adresy, wskaźniki

Celem ćwiczenia jest zapoznanie się ze sposobem deklaracji zmiennych i adresowania pamięci w języku asemblera procesora 8086.

Zagadnienia do przygotowania

- deklaracje i interpretacje zmiennych w języku C
- wskaźniki i tablice, operacje na wskaźnikach
- struktury i unie
- budowa elementarnych programów w języku asemblera
- asemblacja, linkowanie i śledzenie programów w pakiecie Borland C 3.1

Przebieg ćwiczenia

- napisać program podany w załączniku w języku asemblera, np. `lab1.asm`
- przeprowadzić jego kompilację i linkowanie za pomocą pakietu Borland C 3.1:

  `tasm /zi lab1.asm` - asemblacja

  `tlink /v lab1.obj` - linkowanie
- wykonać sprawdzenie poprawności działania programu za pomocą Turbo Debugera.

Warunki zaliczenia ćwiczenia

Zaliczenie ćwiczenia polega na demonstracji działania programu prowadzącemu.

Sprawozdanie

Sprawozdanie powinno zawierać wydruk programu z **obszernymi komentarzami** zawierającymi wartości zmiennych po wykonaniu instrukcji oraz graficzną prezentację zadeklarowanych zmiennych strukturalnych (tablic, struktur i unii).

Literatura

- Kernigham, Ritchie, *Język C*, WNT 1989
- *Dokumentacja pakietu Borland C 3.1*
- Wróbel Eugeniusz, *Asembler 8086/88*
- Wacławek Roland, *ABC Asemblera*
- Scanlon Leo J., *Assembler 8086/8088/80286*
- Kruk Stanisław, *Język Assembler dla początkujących*
- Nawrocki J. R., *Programowanie komputerów IBM PC w języku Asemblera metodą systematyczną*
- Syck Gary, *Turbo Assembler : biblia użytkownika*

Laboratorium: Programowanie niskopoziomowe

Deklaracje zmiennych

```
char c,c1,c2;
int i;
char Tabl[10];
char Napis[] = "asembler";
char *ptr;
struct hilo {char l; char h;};
union  u1 { int i;
            struct hilo c;};
union u1 utab[4];
union u1 *uptr;
```

Instrukcje programu

```
void main(void)
{
  c=1;
  i=-1;
  ptr=Tabl;
  *ptr++=1;
  *ptr++=2;
  c=*((ptr--)-1);
  utab[0].i  =0x2211;
  utab[1].c.l=0x33;
  utab[1].c.h=0x44;
  utab[2].c.l=0x55;
  utab[2].c.h=0x66;
  ptr=(char *)&utab[0];
  i=*(ptr+1);
  ptr = (char *) ((union u1 *)ptr+1);
  c1=*ptr++;
  c2=*ptr++;
  uptr = utab;
  i=(uptr+1)->i;
  uptr++;
  i=(uptr+1)->i;
}
```