תוכניות דוגמא math_s2.c, mathsu2a.asm, math_s1.c, mathsula.asm
math_s3.c, mathsu4a.asm,

כל שלושת התוכניות המשלבות הללו עושות אותו דבר: מקבלות 2 מספרים מהמשתמש, מחשבות את ההפרש ביניהם ומדפיסות את התוצאה על המסך. ההבדל היא בדיוק שהם עובדים בו: הראשונה היא ב-float, השניה היא ב-double והשלישי היא ב- long double.

בכל שלושת התוכניות ההפרש מחושב בפונקציה הכתובה בקובץ אסמבלי טהור בשם math_sub, אך הן נבדלות בהגדרת סוג הפרמטרים ובסוג התוצאה. math_sub תמיד מחזיר את תוצאת החיסור ב-ST(0), כי כאשר פונקצית Turbo C מחזירה תוצאה ממשית היא מוחזרת ב-ST(0) בלי קשר לשאלה אם מדובר ב-float, double או long double.

המימושים של math_sub ב-mathsula.asm שהוא המימוש של ההפרש ב-float וזו ב-mathsu2a.asm שהוא המימוש של ההפרש ב-double ממשות את אותו האלגוריתם. בשני הריצות מדובר במהלך של טעינת 325.89 לתוך ST(0) ולאחר החסרת 542.67 יהיה ב-ST(0) המספר 216.78-.

שני המימושים הללו נבדלים רק בשתי שורות:

ב-mathsula.asm

```
FLD  DWORD PTR [BP+4]
FSUB DWORD PTR [BP+8]
```

ב-mathsula.asm

```
FLD  QWORD PTR [BP+4]
FSUB QWORD PTR [BP+12]
```

לפיכך ההבדל בין 2 המימושים הוא בצורת ההתיחסות לפרמטרים והצורך להביא בחשבון את גודלם במחסנית. זאת משום ששיטת החישוב כאן עקרונית זהה: התוכנית קוראת את המספר הראשון מהמחסנית לתוך ST(0) ומחסירה ממנו את המספר השני ממקומו במחסנית. זאת משם שניתן להחסיר מספרים ממשיים 32 ו-64 בזיכרון ביט מ-ST(0).
כל זה לא נכון אם מדובר חישוב הפרש של מספרים long double. במקרה הזה חייבים לקרוא את שני המספרים לתוך אוגרי המעבד המתמטי ולבצע את ההפרש שם. ואכן המימוש של החיסר בקובץ mathsu4a.c הינו

```
FLD TBYTE PTR [BP+4]
FLD TBYTE PTR [BP+14]
FSUB
```

שים לב שחשוב הסדר של טעינת האופרנדים וש-FSUB ללא אופרנדים עושה POP אוטומטי ומשחרר את ST(1) כפי שהדבר נחוץ. לפיכך המהלך המתרחש בזמן הריצה של התוכנית הינו:

שלב ראשון

ST(0) 325.89
ST(1) ריק

שלב שני

ST(0) 542.78
ST(1) 325.89

שלב שלישי

ST(0) -216.78
ST(1) ריק

נקודות נוספות שיש לשים לב אליהן:

- בכדי שהאסמבלר יכיר בפקודות מכונה של המעבד המתמטי יש לציין הנחיה מתאימה כמו .387, .87, .8087 וכו'.

- אי אפשר לציין .387 לבד בלי .386 קודם ובצורה דומה להנחיות דומות (.87 היא יוצא דופן בכלל הזה).

- התוכניות האסמבלי נכתבו באסמבלי סטנדרטי לא משום שיש צורך מיוחד בכך ולא משום שיש שיקול מיוחד בהקשר של תכנות המעבד המתמטי, אלא בכדי להמחיש גם את הנושא הזה בקורס.

- כאשר משלבים תוכנית אסמבלי סטנדרטי בתוכנית Turbo C חייבים לבחור את שמות הסגמנטים ש-Turbo C בוחר, וחלק מאותם המאפיינים. סגמנט הקוד של תוכנית האסמבלי נקרא _TEXT משום שזהו השם ש-Turbo C בוחר, ומאפיינים נוספים שחייבים להבחר הם PUBLIC ו-'CODE'. הנוהג לכנות את הסגמנט הביצועי בשם (הלא מוצלח לדעת רבים) text segment הוא כנראה נוהג מראשית ימי המחשבים, שקשור כנראה לשיטת העבודה של תכנות באותם ימים.

```
/* math_s1.c -
        Call math_sub (assembler, math co-processor)   */

#include <stdio.h>

extern float math_sub(float u, float v );

void main(void)
{
  float x, y, z;

  puts("Enter two reals:");
  scanf("%f %f", &x, &y);
  z = math_sub(x, y );
  printf("\n%f - %f = %f\n", x, y, z);

} /* main */
```

```
E:\>math_s1
Enter two reals:
325.89  542.67

325.890015 - 542.669983 = -216.779968

E:\>
```

```
;    mathsu1a.asm - implement asembler C-callable procedure
;     float math_sub( float u,    float v ).
;                          [BP+4]      [BP+8]
;
; Result returned in ST
;
     ASSUME CS:_TEXT
;
     PUBLIC _math_sub
;
_TEXT   SEGMENT BYTE PUBLIC 'CODE'
     .386
     .387
_math_sub   PROC NEAR
;
      PUSH BP                          ; Preserve BP register
      MOV  BP,SP
;
      FLD  DWORD PTR [BP+4]           ; Read u into ST
      FSUB DWORD PTR [BP+8]           ; ST = ST - v
;
      POP  BP                          ; Restore BP
      RET                              ; Return to caller
_math_sub   ENDP
_TEXT   ENDS
      END
```

```
/* math_s2.c -
        Call math_sub (assembler, math co-processor)   */

#include <stdio.h>

extern double math_sub(double u, double v );

void main(void)
{
  double x, y, z;

  puts("Enter two reals:");
  scanf("%lf %lf", &x, &y);
  z = math_sub(x, y );
  printf("\n%lf - %lf = %lf\n", x, y, z);

} /* main */
```

```
E:\>math_s2
Enter two reals:
325.89  542.67

325.890000 - 542.670000 = -216.780000

E:\>
```

```
;    mathsu2a.asm - implement asembler C-callable procedure
;     float math_sub( double u,    double  v ).
;                          [BP+4]       [BP+12]
;
; Result returned in ST
;
     ASSUME CS:_TEXT
;
     PUBLIC _math_sub
;
_TEXT  SEGMENT BYTE PUBLIC 'CODE'
     .386
     .387
_math_sub  PROC NEAR
;
      PUSH BP                          ; Preserve BP register
      MOV  BP,SP
;
      FLD  QWORD PTR [BP+4]           ; Read u into ST
      FSUB QWORD PTR [BP+12]           ; ST = ST - v
;
      POP  BP                          ; Restore BP
      RET                              ; Return to caller
_math_sub  ENDP
_TEXT  ENDS
      END
```

```
/* math_s3.c -
        Call math_sub (assembler, math co-processor)  */

#include <stdio.h>

extern long double math_sub(long double u, long double v );

void main(void)
{
  long double x, y, z;

  puts("Enter two reals:");
  scanf("%Lf %Lf", &x, &y);
  z = math_sub(x, y );
  printf("\n%Lf - %Lf = %Lf\n", x, y, z);

} /* main */
```

```
E:\>math_s3
Enter two reals:
325.89  542.67

325.890000 - 542.670000 = -216.780000

E:\>
```

```
;    mathsu3a.asm - implement asembler C-callable procedure
;    long double math_sub(
;        long double u, long double v ).
;           [BP+4]          [BP+14]
;
; Result returned in ST
;
    ASSUME CS:_TEXT
;
    PUBLIC _math_sub
;
_TEXT SEGMENT BYTE PUBLIC 'CODE'
    .386
    .387
_math_sub  PROC NEAR
;
     PUSH BP                        ; Preserve BP register
     MOV  BP,SP
;
     FLD  TBYTE PTR [BP+4]      ; Read u into ST
     FLD  TBYTE PTR [BP+14]     ; ST = v, ST(1) = u
     FSUBP ST(1),ST             ; ST(1) = ST(1) - ST,
                                ; pop (ST = ST(1), ST(1) = Empty)
     POP  BP                      ; Restore BP
     RET                          ; Return to caller
_math_sub  ENDP
_TEXT  ENDS
     END
```

```
;    mathsu4.asm - implement asembler C-callable procedure
;    long double math_sub(
;        long double u, long double v ).
;           [BP+4]         [BP+14]
;
; Result returned in ST
;
    ASSUME CS:_TEXT
;
    PUBLIC _math_sub
;
_TEXT SEGMENT BYTE PUBLIC 'CODE'
    .386
    .387
_math_sub  PROC NEAR
;
     PUSH BP                        ; Preserve BP register
     MOV  BP,SP
;
     FLD  TBYTE PTR [BP+4]      ; Read u into ST
     FLD  TBYTE PTR [BP+14]     ; ST = v, ST(1) = u
     FSUB                       ; ST(1) = ST(1) - ST,
                                ; pop (ST = ST(1), ST(1) = Empty)
     POP  BP                     ; Restore BP
     RET                         ; Return to caller
_math_sub  ENDP
_TEXT  ENDS
     END
```