

ביצוע השוואות במעבד המתמטי

אחד התפקידים של המעבד המתמטי הוא לתמוך בהשוואות בין מספרים ממשיים: כלומר לדעת בין שני מספרים (שלפחות אחד מהם, מן הסתם, ממשי) האם הם שווים, או האם הראשון גדול מהשני וכו'. התהליך אינו לחלוטין פשוט.

לפני שנכנס לפרטים של איך זה עובד ולמה זה נכון, להלן מרשם איך לבצע את השוואה בין שני מספרים.

הדרך המקובלת לבצע השוואה בין שני מספרים הוא כלהלן:

1. הבא לפחות את אחד המספרים לתוך אחד מאוגרי ה-ST(i).

2. השתמש באחד מפקודות ההשוואה (FCOMx, FICOMx)

3. בצע את הפקודת המכונה FSTSW AX

4. בצע את פקודת המכונה SAHF

5. במידה וקיימת אפשרות שהאופרנדים אינם ניתנים להשוואה בדוק זאת על ידי בדיקת ה-PF (על ידי הפקודות JP או JNP).

6. המשך כאילו בצעת את פקודת המכונה CMP על אופרנדים שלמים, אותם אתה מעוניין לפרש כמספרים שלמים חסרי סימן.

לדוגמא, הקוד הבא משווה בין שני משתנים ממשיים בזיכרון Var1 ו-Var2 (32 או 64 ביט, נניח), כאשר הוא מציב לתוך CX את הערך 1 אם $Var1 > Var2$. לצידו ישנו קוד מקביל למצב שבו Var1 ו-Var2 היו משתנים שלמים חסרי סימן 32 ביט:

MOV CX,1

FLD Var1
FCOMP Var2
FSTSW AX
SAHF

JA Skip1
MOV CX,0
Skip1:

MOV CX,1

MOV EDX,Var1
CMP EDX,Var2

JA Skip1
MOV CX,0
Skip1:

שים לב בשימוש ב-JA ולא ב-JG.

בתוך ה-Status Word קיימים ארבע ביטים הקרויים C0, C1, C2, C3.

כאשר מתבצע השוואה, הביטים C0, C2, C3 הם שמכילים את תוצאת ההשוואה.

C2 משקף מצב שבו האיברים שנעשו בהם השוואה אינם ניתנים להשוואה (ערכים מיוחדים של NaN, אינסוף)

C0 ו-C3 מכילים את תוצאות ההשוואה (בהנחה שהאופרנדים היו תקינים).

C3 הוא למעשה ה-Zero Flag של המעבד המתימטי: $C3 = 1$ פירושו שהאופרנדים היו שווים.

$C0 = 1$ אם האופרנד השני (Var2 בדוגמא) גדול מהראשון. האופרנד הראשון הוא בדרך כלל ST(0).

לפיכך אפשר לסכם את משמעות הביטים C0, C2, C3 בטבלה הבאה:

C3	C2	C0	משמעות
0	0	0	אופרנד ראשון < אופרנד שני
0	0	1	אופרנד ראשון > אופרנד שני
1	0	0	אופרנד ראשון == אופרנד שני
1	1	1	אופרנדים בלתי ניתנים להשוואה

בפועל, בדרך כלל נתעלם מהנושא של אופרנדים "בלתי ניתנים להשוואה". בדרך כלל באחריות המתכנת לדאוג לכך שפקודות ההשוואה יופעלו רק על אופרנדים תקינים. כשמצמצמים את הטבלה האחרונה תוך ויתור על ה-PF והאפשרות של בלתי ניתנים להשוואה, מקבלים

C3	C0	משמעות
0	0	אופרנד ראשון < אופרנד שני
0	1	אופרנד ראשון > אופרנד שני
1	0	אופרנד ראשון == אופרנד שני

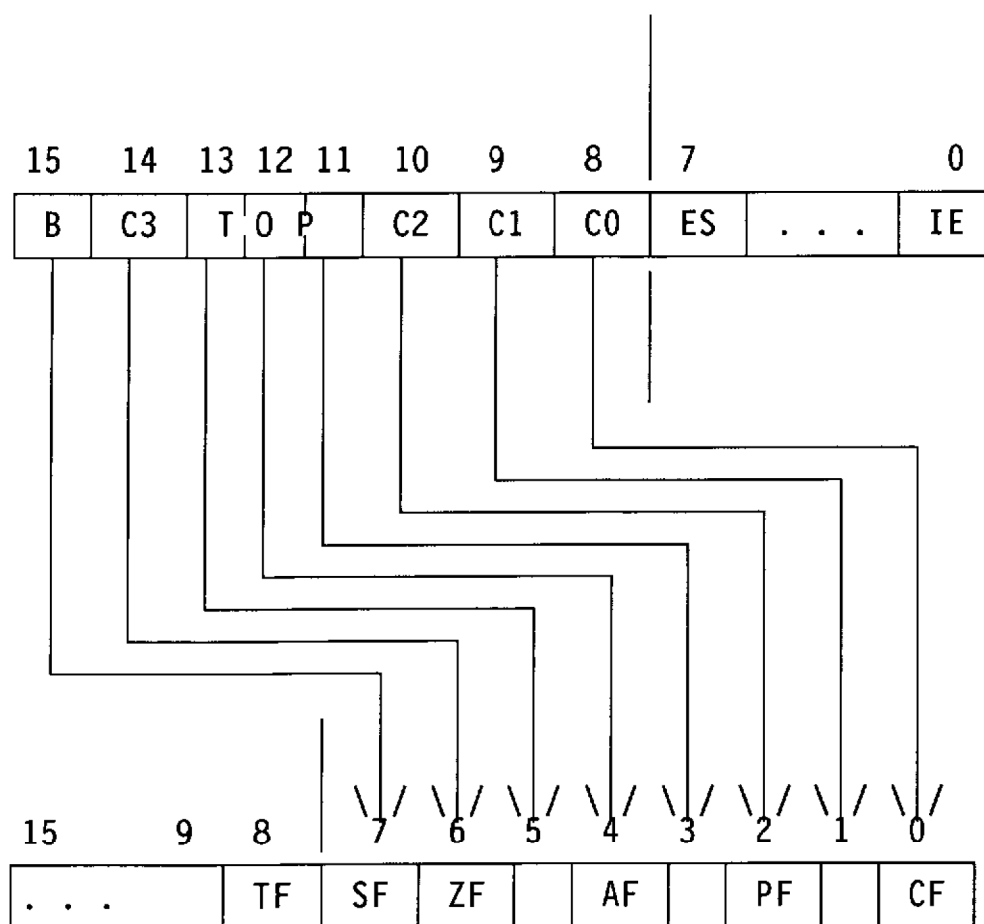
מה שממחיש את תסריט ש-C3 מתנהג כמו ZF ו-C0 מתנהג כמו carry (או יותר נכון borrow) בכל הקשור להשוואה בין מספרים ממשיים, כאילו היו מספרים חסרי סימן.

הפקודה FSTSW AX מעביר את תוכן ה-Status Word לתוך AX.

הפקודה SAHF מעתיקה את AH לתוך החלק התחתון של אוגר הדגלים. במידה ו-AX מכיל את תוכן ה-Status Word, משמעות הדבר יהיה ש-PF יקבל את הערך של C2, ZF יקבל את הערך של C3, ו-CF יקבל את הערך של C0.

המשמעות של פעולת ה-SAHF + FSTSW AX

ה-Status Word של המעבד המתמטי



אוגר הדגלים

כהערות אגב נרסיף את הפרטים הבאים על C0 - C3:

ל-C1 אין תפקיד בהשוואה בין מספרים.

ל-C1 יש מספר תפקידים. במקרים של ביצוע פעולה חוקית שמחייב עיגול (למשל חישוב $1/3$) הערך שלו משקף את סוג העיגול שהתבצע בפועל (עיגול כלפי מעלה $C1 = 1$, כלפי מטה $C1 = 0$).

ל-C1 יש גם תפקיד בפענוח מקרה של Stack Fault (0 במקרה של Underflow, 1 במקרה של Overflow). כלומר תפקידו לאפשר הבדלה בין מצב שניסינו להכניס איבר תשיעי למחסנית (Stack Overflow) נניח ע"י FLD לבין מצב שניסינו לרוקן מהמחסנית יותר איברים משהיו בה (Stack Underflow) נניח ע"י FCOMPP כאשר היה רק ערך אחד במחסנית.

ל-C0, C1, C3 יש גם תפקיד מיוחד בפקודות FPREM ו-FPREM1: הם מכילים את הביטים הכי פחות משמעותיים של התוצאה ($C2$ מקבל 0 בפקודות הללו). לא נכנס לזה כאן.

סיכום

המנגנון מתנהל כך: ביצוע אחת מפקודות ה-FCOM קובע את ערכי ה-C0, C2, C3, הפקודה FSTSW AX מעבירה את הערכים ל-AX והפקודה SAHF מעבירה אותם משם לדגלים $ZF = C3$, $CF = C0$, $PF = C2$. בדיקת PF מאפשר לנו לגלות שהבדיקה היתה בין אופרנדים שאינם ניתנים להשוואה. במידה שזה לא היה המקרה (או שמראש אין אפשרות כזו) $ZF = 1$ משקף מצב של שיוון ובמידה ו- $ZF = 0$ אזי CF משקף מי מהאופרנדים גדול יותר: $CF = 0$ הראשון, $CF = 1$ השני. קצת אנליזה תראה שזה בדיוק המצב של השוואת בין מספרים חסרי סימן, כאשר CF משקף אפשרות של Borrow של חיסור חסרי סימן.

כמובן שאפשר גם לבדוק את ערכי הביטים ישירות לאחר הפקודה FSTSW AX ע"י הפקודות TEST או AND. אחת מתוכניות הדוגמא שבהמשך מממשת את האפשרות הזו.

תוכניות דוגמא ccomp1.c, mc1.asm, ccomp2.c mc2.asm, ccomp3.c mc3.asm
mc3.asm

מטרת התוכניות המשולבות הזו להמחיש את נושא השוואת 2 מספרים ממשיים במעבד המתמטי.

התוכניות הראשית הכתובה ב-C מקבלת 2 מספרים מהמשתמש כפרמטרים לתוכנית ראשית וקוראת לפונקציה האסמבלי comp להשוות כיניהם. מחזירה תוצאה שלמה 0 אם שניהם שווים, 1 אם הפרמטר הראשון גדול מהשני ו-1- אם הפרמטר השני גדול מהראשון. ההנחה היא ששני המספרים הממשיים תקינים, לא מכילים בחשבון את האפשרות של "לא ניתנים להשוואה". הזוג mc2.asm ו-ccomp1.c מימש את החישוב ב-float, הזוג mc3.asm ו-ccomp2.c מימש את החישוב ב-double ו-ccomp3.c mc3.asm מימש את החישוב ב-long double.

אשר למימוש של עצמם, הם נעשים לפי המרשם של השוואות מספרים במעבד המתמטי שתואר קודם. התוכניות mc1.asm ו-mc2.asm מנצלות את העובדה שניתן בפקודה FCOMP להשוות מספרים ממשיים 32 ו-64 ביט ישירות מהזכרון ואילו mc3.asm, מאחר והיא משווה 2 מספרים 80 ביט, חייבת לטעון את שניהם לאוגרי המעבד ומבצעת שם את הפקודה FCOMPP. שימו לב לסדר הטעינה של המספרים ב-mc3.asm. על מנת לשמר ב-mc3.asm את אותה פקודת הסתעפות (JBE) כמו ב-mc1.asm ו-mc2.asm, התוכנית טוענת למעבד קודם את המספר השני ורק אחריו את המספר הראשון. כתוכנית הדבר בא לידי ביטוי בשורות

```
FLD TBYTE PTR [BP+14] ; ST(0) = p1
FLD TBYTE PTR [BP+4]
FCOMPP ; Compare p1 with p2, pop twice
```

זאת משום שהפקודה FCOMPP משווה את ST(0) לעומת ST(1).

מענין לעשות אנלוגיה למצב שבו נניח היינו רוצים לממש רוטינה דומה המחזירה אותה תשובה נניח אם היינו משווים 2 מספרים אי שליליים 32 ביט.

השוואה במעבד המתימטי

אנלוגיה לשלמים

~~FLD DWORD PTR [BP+4]~~
FCOMP DWORD PTR [BP+8]
FSTSW AX
SAHF



MOV EAX, [BP+4]
CMP EAX, [BP+8]

JBE Not_Above
MOV AX, 1
JMP SHORT Finish
Not_Above:
JE Equal
MOV AX, -1
JMP SHORT Finish

Equal:
MOV AX, 0

Finish:
POP BP
RET

JBE Not_Above
MOV AX, 1
JMP SHORT Finish
Not_Above:
JE Equal
MOV AX, -1
JMP SHORT Finish

Equal:
MOV AX, 0

Finish:
POP BP
RET

הערה: "JMP SHORT" מנחה את האסמבלר לבחור את הגרסה של -128 + 127 של JMP במקרה של הסתעפות קדימה זה מיעל במשהו את הקובץ הכינארי שנוצר. זה לא קשור בהכרח למעבד המתימטי. לא נכנס לזה כאן.

```

/* ccomp1.c - compare real numbers */

#include <stdio.h>
#include <stdlib.h>

extern int comp( float p1, float p2 );

void main(int argc, char *argv[])
{
    float x, y;
    int cmp_flag;

    if (argc < 3)
    {
        fprintf(stderr, "Usage: ccomp1 real1 real2\n");
        exit(1);
    }

    sscanf(argv[1], "%f", &x);
    sscanf(argv[2], "%f", &y);

    cmp_flag = comp(x, y);

    if ( cmp_flag == 0 )
        printf("%f == %f\n\n", x, y);
    else
        if ( cmp_flag > 0 )
            printf("%f > %f\n\n", x, y);
        else
            if ( cmp_flag < 0 )
                printf("%f < %f\n\n", x, y);
} /* main */

```

```

E:\>tcc ccomp1.c mcl.asm
Turbo C++ Version 3.00 Copyright (c) 1992 Borland International
ccomp1.c:
mcl.asm:
Turbo Assembler Version 3.1 Copyright (c) 1988, 1992 Borland
International

```

```

Assembling file:    mcl.ASM
Error messages:     None
Warning messages:   None
Passes:             1
Remaining memory:   398k

```

```

Turbo Link Version 5.0 Copyright (c) 1992 Borland International

```

```

    Available memory 4104476

```

```

E:\>CCOMP1.EXE 432.65 988.22
432.649994 < 988.219971

```

```

E:\>

```

220

```

; mcl.asm - Compare within the math coprocessor
;
;      int _comp( float p1, float p2)
;                  [BP+4]   [BP+8]
;
;      If p1 = p2,      Return 0
;      If p1 > p2,      Return 1
;      If p1 < p2,      Return -1
;
; FCOMP = Compare and Pop
; Return  C3 C2 C1 C0 Meaning
;          0  0  ?  0   ST > Source
;          0  0  ?  1   ST < Source
;          1  0  ?  0   ST == Source
;          1  1  ?  1   ST is not comparable to source
;
; SAHF - Store AH into flags
;
;      Status Word
;      -----
;      15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
;      ---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
;      | B | C3 | TOP | C2 | C1 | C0 | ES | SF | PE | UE | OE | ZE | DE | IE |
;      ---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
;
PUBLIC _comp
;
.MODEL SMALL
;
.CODE
.386
.387
_comp PROC NEAR
    PUSH    BP
    MOV     BP, SP
    FLD     DWORD PTR [BP+4] ; ST(0) = p1
    FCOMP   DWORD PTR [BP+8] ; Compare ST with p2, pop
    FSTSW   AX               ; Store comparison result in AX
;
    SAHF    ; Store C3, C2, C0 in Flags register (ZF, PF, CF)
;
; Proceed as we did CMP on two unsigned integer operands
;
;           ; Was ST > ST(1) ( p1 > p2 ) ?
JBE     Not_Above ; No, more tests
    MOV     AX, 1 ; Yes, return 1
    JMP     SHORT Finish
Not_Above: ; ST <= ST(1)
;           ; Was ST < ST(1) ( p1 < p2 ) ?
JE      Equal ; No, ST == ST(1)
    MOV     AX, -1 ; Yes, return 0
    JMP     SHORT Finish
;
Equal:
    MOV     AX, 0 ; ST == ST(1) or incomparable
;
Finish:
    POP     BP
    RET
ENDP _comp
;
;
END

```

```

/* ccomp2.c - compare real numbers */

#include <stdio.h>
#include <stdlib.h>

extern int comp( double p1, double p2 );

void main(int argc, char *argv[])
{
    double x, y;
    int cmp_flag;

    if (argc < 3)
    {
        fprintf(stderr, "Usage: ccomp1 real1 real2\n");
        exit(1);
    }

    sscanf(argv[1], "%lf", &x);
    sscanf(argv[2], "%lf", &y);

    cmp_flag = comp(x, y);

    if ( cmp_flag == 0 )
        printf("%lf == %lf\n\n", x, y);
    else
        if ( cmp_flag > 0 )
            printf("%lf > %lf\n\n", x, y);
        else
            if ( cmp_flag < 0 )
                printf("%lf < %lf\n\n", x, y);
} /* main */

```

```

E:\>tcc ccomp2.c mc2.asm
Turbo C++ Version 3.00 Copyright (c) 1992 Borland International
ccomp2.c:
mc2.asm:
Turbo Assembler Version 3.1 Copyright (c) 1988, 1992 Borland
International

```

```

Assembling file:    mc2.ASM
Error messages:     None
Warning messages:   None
Passes:             1
Remaining memory:   398k

```

```

Turbo Link Version 5.0 Copyright (c) 1992 Borland International

```

```

    Available memory 4103660

```

```

E:\>ccomp2.exe 321.2 -400.9
321.200000 > -400.900000

```

```

E:\>

```

222

```

; mc2.asm - Compare within the math coprocessor
;
;      int _comp( double p1, double p2)
;                  [BP+4]   [BP+12]
;
; If p1 = p2,      Return 0
; If p1 > p2,      Return 1
; If p1 < p2,      Return -1
;
; FCOMP = Compare and Pop
; Return  C3 C2 C1 C0  Meaning
;         0  0  ?  0    ST > Source
;         0  0  ?  1    ST < Source
;         1  0  ?  0    ST == Source
;         1  1  ?  1    ST is not comparable to source
;
; SAHF - Store AH into flags
;
; Status Word
; -----
; 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
; -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
; | B | C3 | TOP | C2 | C1 | C0 | ES | SF | PE | UE | OE | ZE | DE | IE |
; -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
;
; .MODEL SMALL
; .CODE
; .386
; .387
; _comp PROC NEAR
; PUBLIC _comp
; PUSH BP
; MOV BP, SP
; FLD QWORD PTR [BP+4] ; ST(0) = p1
; FCOMP QWORD PTR [BP+12] ; Compare ST with p2, pop
; FSTSW AX ; Store comparison result in AX
;
; SAHF ; Store C3, C2, C0 in Flags register (ZF, PF, CF)
;
; ; Proceed as we did CMP on two unsigned integer operands
;
; ; Was ST > ST(1) ( p1 > p2 ) ?
; JBE Not_Above ; No, more tests
; MOV AX, 1 ; Yes, return 1
; JMP SHORT Finish
;
; Not_Above: ; ST <= ST(1)
; ; Was ST < ST(1) ( p1 < p2 )?
; JE Equal ; No, ST == ST(1)
; MOV AX, -1 ; Yes, return 0
; JMP SHORT Finish
;
; Equal:
; MOV AX, 0 ; ST == ST(1) or incomparable
;
; Finish:
; POP BP
; RET
; ENDP _comp
;
;
; END

```

```

; mc2.asm - Compare within the math coprocessor
;
;      int _comp( double p1, double p2)
;                  [BP+4]   [BP+12]
;
;      If p1 = p2,   Return 0
;      If p1 > p2,   Return 1
;      If p1 < p2,   Return -1
;
; FCOMP = Compare and Pop
; Return  C3  C2  C1  C0  Meaning
;         0   0  ?   0    ST > Source
;         0   0  ?   1    ST < Source
;         1   0  ?   0    ST == Source
;         1   1  ?   1    ST is not comparable to source
;
; SAHF - Store AH into flags
;
; Status Word
; -----
; 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
; ---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
; | B | C3 | TOP | C2 | C1 | C0 | ES | SF | PE | UE | OE | ZE | DE | IE |
; ---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
;
; .MODEL SMALL
; .CODE
; .386
; .387
; _comp PROC NEAR
; PUBLIC _comp
; PUSH BP
; MOV BP, SP
; FLD QWORD PTR [BP+4] ; ST(0) = p1
; FCOMP QWORD PTR [BP+12] ; Compare ST with p2, pop
; FSTSW AX ; Store comparison result in AX
;
; SAHF ; Store C3, C2, C0 in Flags register (ZF, PF, CF)
;
; ; Proceed as we did CMP on two unsigned integer operands
;
; ; Was ST > ST(1) ( p1 > p2 ) ?
; JBE Not_Above ; No, more tests
; MOV AX, 1 ; Yes, return 1
; JMP SHORT Finish
;
; Not_Above: ; ST <= ST(1)
; ; Was ST < ST(1) ( p1 < p2 )?
; JE Equal ; No, ST == ST(1)
; MOV AX, -1 ; Yes, return 0
; JMP SHORT Finish
;
; Equal:
; MOV AX, 0 ; ST == ST(1) or incomparable
;
; Finish:
; POP BP
; RET
; ENDP _comp
;
;
; END

```

```

/* ccomp3.c - compare real numbers */

#include <stdio.h>
#include <stdlib.h>

extern int comp( long double p1, long double p2 );

void main(int argc, char *argv[])
{
    long double x, y;
    int cmp_flag;

    if (argc < 3)
    {
        fprintf(stderr, "Usage: ccomp1 real1 real2\n");
        exit(1);
    }

    sscanf(argv[1], "%Lf", &x);
    sscanf(argv[2], "%Lf", &y);

    cmp_flag = comp(x, y);

    if ( cmp_flag == 0 )
        printf("%Lf == %Lf\n\n", x, y);
    else
        if ( cmp_flag > 0 )
            printf("%Lf > %Lf\n\n", x, y);
        else
            if ( cmp_flag < 0 )
                printf("%Lf < %Lf\n\n", x, y);
} /* main */

```

```

E:\users\eytan\asm\braude>tcc -v ccomp3.c mc3.asm
Turbo C++ Version 3.00 Copyright (c) 1992 Borland International
ccomp3.c:
mc3.asm:
Turbo Assembler Version 3.1 Copyright (c) 1988, 1992 Borland
International

```

```

Assembling file:    mc3.ASM
Error messages:     None
Warning messages:   None
Passes:             1
Remaining memory:   429k

```

```

Turbo Link Version 5.0 Copyright (c) 1992 Borland International

    Available memory 4138012

```

```

E:\>CCOMP3.EXE 654.6 32.87
654.600000 > 32.870000

```

```

E:\>

```

225

```

; mc3.asm - Compare within the math coprocessor
;
;      int _comp( long double p1, long double p2)
;                [BP+4]          [BP+14]
;
;  If p1 = p2,      Return  0
;  If p1 > p2,      Return  1
;  If p1 < p2,      Return -1
;
;FCOMP = Compare and Pop
;Return  C3  C2  C1  C0  Meaning
;         0   0   ?   0    ST > Source
;         0   0   ?   1    ST < Source
;         1   0   ?   0    ST == Source
;         1   1   ?   1    ST is not comparable to source
;
; SAHF - Store AH into flags
;
;  Status Word
;  -----
;  15  14   13 12 11  10    9    8    7    6    5    4    3    2    1    0
;  -+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
;  | B | C3 | TOP | C2 | C1 | C0 | ES | SF | PE | UE | OE | ZE | DE | IE |
;  -+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
;
;
;
.MODEL SMALL
.CODE
.386
.387
_comp PROC NEAR
PUBLIC _comp
PUSH BP
MOV BP,SP
FLD TBYTE PTR [BP+14] ; ST(0) = p2
FLD TBYTE PTR [BP+4]
FCOMPP ; Compare p1 with p2, pop twice
FSTSW AX ; Store comparison result in AX
;
SAHF ; Store C3, C2, C0 in Flags register (ZF, PF, CF)
;
; Proceed as we did CMP on two unsigned integer operands
;
; Was ST > ST(1) ( p1 > p2 ) ?
JBE Not_Above ; No, more tests
MOV AX,1 ; Yes, return 1
JMP SHORT Finish
Not_Above: ; ST <= ST(1)
; Was ST < ST(1) ( p1 < p2 )?
JE Equal ; No, ST == ST(1)
MOV AX,-1 ; Yes, return 0
JMP SHORT Finish
;
Equal:
MOV AX,0 ; ST == ST(1) or incomparable
;
Finish:
POP BP
RET
ENDP _comp
;
;
END

```