X86 Assembly Language
and
C Fundamentals

Chapter 5

Data Transfer Instructions

X86 Code Figures

Page 200, Figure 5.4

```
;swap_bytes.asm
;-------------------------------------------------------------
.STACK
;-------------------------------------------------------------
.DATA
    TEMP        DB ?
;$ sign is a delimiter meaning end of string
    RSLT        DB 0DH, 0AH, 'BL =  , BH =  $'
;-------------------------------------------------------------
.CODE
BEGIN   PROC FAR

;set up pgm ds
      MOV     AX, @DATA    ;place the .DATA addr in ax
      MOV     DS, AX       ;set up data seg addr for this pgm

;assign values to bl and bh
      MOV     BL, 'A'
      MOV     BH, 'B'

;store bl in temp area before swapping
      MOV     TEMP, BL

;swap registers
      MOV     BL, BH
      MOV     BH, TEMP

;move registers to result area for display
      MOV     RSLT + 7, BL
      MOV     RSLT + 15, BH

;print result
      MOV     AH, 09H         ;display string
      MOV     DX, OFFSET RSLT ;rslt addr must be in dx
      INT     21H             ;a dos interrupt that uses
                              ;a fctn code in ah

BEGIN ENDP
      END     BEGIN           ;start pgm at begin
```

(a)

```
BL = B, BH = A
```

(b)

Page 202, Figure 5.5

```
//swap_bytes.cpp
//swap bytes in two GPRs

#include "stdafx.h"
char main (void)

{
   char temp;
   char rslt1, rslt2;

//switch to assembly
      _asm
      {
         MOV   BL, 'A'
         MOV   BH, 'B'
//swap bytes
         MOV   temp, BL
         MOV   rslt1, BH
         MOV   BH, temp
         MOV   rslt2, BH
      }

//print result
   printf ("BL = %c, BH = %c\n", rslt1, rslt2);
   return 0;
}
```

(a)

```
BL = B, BH = A
Press any key to continue . . . _(b)
```

Page 203, Figure 5.5

```
//movsx_movzx2.cpp
//move with sign/zero extension

#include "stdafx.h"
int main (void)

{
   int rslt1, rslt2;

//switch to assembly
      _asm
      {
         MOV   AL, 0F5H
         MOVSX EAX, AL
         MOV   rslt1, EAX

         MOV   BL, 0F5H
         MOVZX EBX, AL
         MOV   rslt2, EBX
      }
   printf ("Result = %X, %X\n", rslt1, rslt2);

   return 0;
}
```

(a)

```
Result = FFFFFFF5, F5
Press any key to continue . . . _
```

(b)

Page 205, Figure 5.7

```
//mov_cond.cpp
//uses cmovae (above or equal); cf = 0.
//If user-entered x is unsigned above or equal to
//user-entered y, then print x; otherwise, print y.
//If integers are equal, then print x

#include "stdafx.h"

int main (void)
{
//define and initialize variables
   int   x, y, rslt;

   printf ("Enter two integers: \n");
   scanf ("%d %d", &x, &y);

//switch to assembly
      _asm
      {
         MOV   EAX, x
         MOV   EBX, y

//to move ebx to result if conditional move fails
         MOV   EDX, EBX

         CMP   EAX, EBX    //set flags

//if eax >= ebx, move eax to rslt;
//otherwise, move ebx to result
         CMOVAE EDX, EAX
         MOV    rslt, EDX
      }

   printf ("Result = %d\n\n", rslt);

   return 0;
}
```

(a)

```
Enter two integers:
15 10
Result = 15

Press any key to continue . . . _
------------------------------------------------------------
Enter two integers:
10 15
Result = 15

Press any key to continue . . . _
------------------------------------------------------------
Enter two integers:
10 10
Result = 10

Press any key to continue . . . _
```

(b)

Page 210, Figure 5.12

```
;xchg_2_characters.asm
.STACK

.DATA
PARLST  LABEL BYTE
MAXLEN  DB 5
ACTLEN  DB ?
OPFLD   DB 5 DUP(?
PRMPT DB 0DH, 0AH, 'Enter two characters: $'
RSLT1 DB 0DH, 0AH, 'Before exchange =      $'
RSLT2 DB 0DH, 0AH, 'After exchange =       $'

.CODE
BEGIN PROC FAR
;set up pgm ds
      MOV     AX, @DATA
      MOV     DS, AX

;read prompt
      MOV     AH, 09H         ;display string
      LEA     DX, PRMPT       ;load addr of prmpt
      INT     21H             ;dos interrupt

;kybd rtn to enter characters
      MOV     AH, 0AH         ;buffered kybd input
      LEA     DX, PARLST      ;load addr of parlst
      INT     21H             ;dos interrupt

;store characters from opfld to bl and bh
      MOV     BL, OPFLD       ;get 1st char, store in bl
      MOV     BH, OPFLD + 1   ;get 2nd char, store in bh

;display original characters
      MOV     RSLT1 + 20, BL
      MOV     RSLT1 + 22, BH

      MOV     AH, 09H
      LEA     DX, RSLT1
      INT     21H
                                  //continued on next page
```

(a)

```
      XCHG    BL, BH

;display swapped characters
      MOV     RSLT2 + 19, BL
      MOV     RSLT2 + 21, BH

      MOV     AH, 09H
      LEA     DX, RSLT2
      INT     21H

BEGIN ENDP
      END     BEGIN
```

```
Enter two characters: 12
Before exchange = 1 2
After exchange = 2 1
------------------------------------------------------------
Enter two characters: ab
Before exchange = a b
After exchange = b a
```

(b)

Page 212, Figure 5.13

```
//xchg_2_numb.cpp
//exchange two user-entered integers

#include "stdafx.h"
int main (void)
{
   intx, y, rslt1, rslt2;     //define variables
   printf ("Enter two integers: ");
   scanf ("%d %d", &x, &y);

//switch to assembly
      _asm
      {
         MOV   EAX, x
         MOV   EBX, y
         XCHG  EAX, EBX
         MOV   rslt1, EAX
         MOV   rslt2, EBX
      }
   printf ("Result = %d %d\n\n", rslt1, rslt2);

   return 0;
}
```

(a)

```
Enter two integers: 11 22
Result = 22 11

Press any key to continue . . . _
------------------------------------------------------------
Enter two integers: 200 400
Result = 400 200

Press any key to continue . . . _
```

(b)

Page 214, Figure 5.15

```
//byte_swap.cpp
//swap bytes to convert between little-endian and big-endian

#include "stdafx.h"

int main (void)
{
   //define variables
   int   x, rslt;

   printf ("Enter an integer: \n");
   scanf ("%X", &x);

//switch to assembly
      _asm
      {
         MOV   EAX, x
         BSWAP EAX
         MOV   rslt, EAX
      }

   printf ("\nResult = %X\n\n", rslt);

   return 0;
}
```

(a)

```
Enter an integer:
FF00FF00

Result = FF00FF

Press any key to continue . . . _
------------------------------------------------------------
Enter an integer:
11221122

Result = 22112211

Press any key to continue . . . _
------------------------------------------------------------
Enter an integer:
12345678

Result = 78563412

Press any key to continue . . . _
------------------------------------------------------------
Enter an integer:
-1

Result = FFFFFFFF

Press any key to continue . . . _
------------------------------------------------------------
Enter an integer:
aaaabbbb

Result = BBBBAAAA

Press any key to continue . . . _
```

(b)

Page 216, Figure 5.16

```
//xchg_add.cpp
//exchange and add two general-purpose registers

#include "stdafx.h"

int main (void)
{
//define variables
   int   x, y, rslt1, rslt2;

   printf ("Enter two integers: \n");
   scanf ("%d %d", &x, &y);

//switch to assembly
      _asm
      {
         MOV   EAX, x
         MOV   EBX, y

         XADD  EAX, EBX       //swap EAX and EBX
                              //sum is stored in EAX (dst)
         MOV   rslt1, EAX     //move sum to rslt1
         MOV   rslt2, EBX     //move original EAX to rslt2
      }

   printf ("\nSum = %d\nOriginal EAX = %d\n\n",
      rslt1, rslt2);

   return 0;
}
```

(a)

```
Enter two integers:
40 25

Sum = 65
Original EAX = 40

Press any key to continue . . . _
------------------------------------------------------------
Enter two integers:
672 538

Sum = 1210
Original EAX = 672

Press any key to continue . . . _
```

(b)

Page 218, Figure 5.18

```
//comp_xchg.cpp
//compare and exchange registers based
//on the results of the comparison

#include "stdafx.h"

int main (void)
{
   //define variables
   int   eax_reg, ebx_reg, edx_reg, equal, not_equal;

   printf ("Enter integers for EAX, EBX_dst, EDX_src: \n");
   scanf ("%d %d %d", &eax_reg, &ebx_reg, &edx_reg);

//switch to assembly
      _asm
      {
         MOV      EAX, eax_reg
         MOV      EBX, ebx_reg   //EBX is dst register
         MOV      EDX, edx_reg   //EDX is src register

         CMPXCHG  EBX, EDX       //if EAX = EBX, EDX --> EBX
                                 //if EAX != EBX, EBX --> EAX

         MOV      equal, EBX
         MOV      not_equal, EAX
      }

   printf ("\nEBX = %d\nEAX = %d\n\n", equal, not_equal);

   return 0;
}
```

(a)

```
Enter integers for EAX, EBX_dst, EDX_src:
130 130 140

EBX = 140
EAX = 130

Press any key to continue . . . _
------------------------------------------------------------
Enter integers for EAX, EBX_dst, EDX_src:
120 100 150

EBX = 100
EAX = 100

Press any key to continue . . . _
------------------------------------------------------------
Enter integers for EAX, EBX_dst, EDX_src:
-1 4294967295 20              //4294967295 is -1 (FFFFFFFF)

EBX = 20
EAX = -1

Press any key to continue . . . _
------------------------------------------------------------
Enter integers for EAX, EBX_dst, EDX_src:
4294967295 -1 30

EBX = 30
EAX = -1

Press any key to continue . . . _
------------------------------------------------------------
Enter integers for EAX, EBX_dst, EDX_src:
4294967295 20 55

EBX = 20
EAX = 20

Press any key to continue . . . _
```

(b)

Page 222, Figure 5.20

```
;translate.asm

;Enter an ascii character that corresponds to a 4-bit
;hexadecimal number that is used to generate a 4-bit
;Gray code number 0 - 9, A - F

.STACK

.DATA
PARLST  LABEL BYTE
MAXLEN  DB 5
ACTLEN  DB ?
OPFLD   DB 5 DUP(0)

;define ascii translation table of 256 bytes
XLTBL   DB 30H, 31H, 33H, 32H   ;0, 1, 3, 2
        DB 36H, 37H, 35H, 34H   ;6, 7, 5, 4
        DB 43H, 44H, 46H, 45H   ;12(C), 13(D), 15(F), 14(E)
        DB 41H, 42H, 39H, 38H   ;10(A), 11(B), 9, 8
        DB 240 DUP(30H)         ;0

PRMPT   DB 0DH, 0AH, 'Enter an ascii character: $'
RSLT    DB 0DH, 0AH, 'Gray =       $'

.CODE
BEGIN   PROC FAR

;set up pgm ds
        MOV     AX, @DATA
        MOV     DS, AX

;put addr of translation table xltbl in bx
        LEA     BX, XLTBL

;read prompt
        MOV     AH, 09H     ;display string
        LEA     DX, PRMPT   ;load addr of prmpt
        INT     21H         ;dos interrupt

;kybd rtn to enter characters
        MOV     AH, 0AH     ;buffered kybd input
        LEA     DX, PARLST  ;load addr of parlst
        INT     21H         ;dos interrupt
```

(a)

//continued on next page

```
;store number from opfld in al and translate
        MOV     AL, OPFLD
        AND     AL, 0FH     ;remove ascii bias

        XLATB               ;translate to ascii

        MOV     RSLT+9, AL  ;move gray number to rslt

;display gray number
        MOV     AH, 09H     ;display string
        LEA     DX, RSLT    ;load addr of rslt field
        INT     21H         ;dos interrupt

BEGIN   ENDP
        END     BEGIN
```

```
Enter an ascii character: 3
Gray = 2
------------------------------------------------------------
Enter an ascii character: 5
Gray = 7
------------------------------------------------------------
Enter an ascii character: 8
Gray = C
------------------------------------------------------------
Enter an ascii character: 9
Gray = D
```

(b)

Page 224, Figure 5.21

```
//translate.cpp
//Convert from binary to gray code.

#include "stdafx.h"
int main (void)
{
//define array for gray code
   int gray[256] = {0x00, 0x01, 0x03, 0x02,
                    0x06, 0x07, 0x05, 0x04,
                    0x0c, 0x0d, 0x0f, 0x0e,
                    0x0a, 0x0b, 0x09, 0x08};
   int x, rslt;

   printf ("Enter an integer 0-15 that represents
            a binary number: \n");
   scanf ("%d", &x);

   printf ("Binary = %d, Gray = %x\n\n", x, gray[x]);

   return 0;
}
```

(a)

```
Enter an integer 0-15 that represents a binary number:
13
Binary = 13, Gray = b

Press any key to continue . . . _
------------------------------------------------------------
Enter an integer 0-15 that represents a binary number:
2
Binary = 2, Gray = 3

Press any key to continue . . . _
------------------------------------------------------------
Enter an integer 0-15 that represents a binary number:
15
Binary = 15, Gray = 8

Press any key to continue . . . _
------------------------------------------------------------
Enter an integer 0-15 that represents a binary number:
10
Binary = 10, Gray = f

Press any key to continue . . . _
------------------------------------------------------------
Enter an integer 0-15 that represents a binary number:
11
Binary = 11, Gray = e

Press any key to continue . . . _
```

(b)

Page 227, Figure 5.23

```
//convert.cpp
//convert doubleword to quadword (CDQ)

#include "stdafx.h"
int main (void)

{
//define variables
   int   eax_reg, edx_reg;

   printf ("Enter an integer: \n");
   scanf ("%x", &eax_reg);

//switch to assembly
      _asm
      {
         MOV   EAX, eax_reg
         CDQ
         MOV   eax_reg, EAX
         MOV   edx_reg, EDX
      }

   printf ("\nEDX = %X, EAX = %X\n\n", edx_reg, eax_reg);

   return 0;
}
```

(a)

```
Enter an integer:
F0F0F0F0

EDX = FFFFFFFF, EAX = F0F0F0F0

Press any key to continue . . . _
------------------------------------------------------------
Enter an integer:
ffffffff

EDX = FFFFFFFF, EAX = FFFFFFFF

Press any key to continue . . . _
------------------------------------------------------------
Enter an integer:
3F

EDX = 0, EAX = 3F

Press any key to continue . . . _
------------------------------------------------------------
Enter an integer:
63

EDX = 0, EAX = 63

Press any key to continue . . . _
------------------------------------------------------------
Enter an integer:
80000000

EDX = FFFFFFFF, EAX = 80000000

Press any key to continue . . . _
------------------------------------------------------------
Enter an integer:
7fffffff

EDX = 0, EAX = 7FFFFFFF

Press any key to continue . . . _
```

(b)