Using Assembly Language With C/C++
Introduction

- This chapter develops the idea of mixing C/C++ and assembly language.
- This text uses Microsoft Visual C/C++ Express, but programs can often be adapted to any version of C/C++, as long as it is standard ANSI (American National Standards Institute) format C/C++.
- If you want, you can use C/C++ to enter and execute all the programming applications in this text.
USING ASSEMBLY LANGUAGE WITH C++ FOR 16-BIT DOS APPLICATIONS

• Use software to build 16-bit applications when attempting any programs in this section.
  – if you build a 32-bit application and attempt the DOS INT 21H function, the program will crash
• To build a 16-bit **DOS** application, you will need the legacy 16-bit compiler
• Programs are generated using Notepad or DOS Edit.
Basic Rules and Simple Programs

• Before assembly language code can be placed in a C/C++ program, some rules must be learned.

• Example 1 shows how to place assembly code inside an assembly language block within a short C/C++ program.

• Assembly code in this example is placed in the _asm block.

• Labels are used as illustrated by the label big.
• Extremely important to use lowercase characters for any inline assembly code.
  – if you use uppercase, you will find some assembly language commands and registers are reserved or defined words in C/C++

• Example 1 reads one character from the console keyboard, and then filters it through assembly language so that only the numbers 0 through 9 are sent back to the video display.

• It shows how to set up and use some simple programming constructs in C/C++. 
Example 1

// Accepts and displays one character of 1 through 9, all others are ignored.
void main(void) {
    _asm {
        mov ah,8
        ;read key no echo
        int 21h
        cmp al,‘0’
        ;filter key code
        jb big
        cmp al,‘9’
        ja big
        mov dl,al
        ;echo 0 – 9
        mov ah,2
        int 21h
        big:
    }
}
• Note that AX, BX, CX, DX, and ES registers are never used by Microsoft C/C++. 
  – these might be considered **scratchpad** registers, available to use with assembly language

• If you wish to use any of the other registers, make sure that you save them with a PUSH before they are used and restore them with a POP afterwards.
  – if you fail to save registers used by a program, the program may not function correctly and can crash the computer
Example 2

```c
void main(void) {
    char a, b;
    _asm {
        mov ah,1 ;read first digit
        int 21h
        mov a,al
        mov ah,1 ;read a + sign
        int 21h
        cmp al,'+' jne end1 ;if not plus
        mov ah,2 ;display =
        mov dl,'='
        int 21h
        mov ah,0
        mov al,a
        mov al,a ;generate sum
        add al,b
        aaa
        cmp ah,'0' jne down ;ASCII adjust for addition
        add ax,3030h
        cmp ah,'0' je down
    }
}
```
Example 2 contd...

```assembly
push ax
    ;display 10's position
mov dl,ah
mov ah,2
int 21h
int 21h
pop ax
pop ax
down: mov dl,al
    ;display units position
mov ah,2
mov ah,2
int 21h
int 21h
end1:
```
What Cannot Be Used from MASM Inside an _asm Block

- MASM contains features, such as conditional commands (.IF, .WHILE, .REPEAT, etc.)
- The inline assembler does not include the conditional commands from MASM.
  - nor does it include the MACRO feature found in the assembler
- Data allocation with the inline assembler is handled by C.
Using Character Strings

- Example 3 illustrates a simple program that uses a character string defined with C and displays it so that each word is listed on a separate line.
  - notice the blend of both C statements and assembly language statements

- The WHILE statement repeats the assembly language commands until the null (00H) is discovered at the end of the character string.
• For each space, the program displays a carriage return/line feed combination.
• This causes each word in the string to be displayed on a separate line.
// Example that displays
// showing one word per line

void main(void)
{
  char strings[] = “This is my first
test application using _asm. \n”;
  int sc = -1;
  while (strings[++sc] != ‘\0’)
  {
    _asm
    {
      push si
      mov si,sc
      ;get pointer
      mov dl,strings[si]
      ;get character

      cmp dl,’ ’
      ;if not space
      jne next
      mov ah,2
      ;display new line
      mov dl,10
      int 21h
      mov dl,13
      next: mov ah,2
      ;display character
      int 21h
      pop si
    }
  }
}
Example 4

// A program illustrating an assembly language procedure that displays C language character strings
char string1[] = "This is my first test program using \_asm."
char string2[] = "This is the second line in this program."
char string3[] = "This is the third."
void main(void)
{
    Str (string1);
    Str (string2);
    Str (string3);
}
Str (char *string_adr)
{
    _asm
    {
        mov bx,string_adr
        ;get address of string
        mov ah,2
        top:
        mov dl,[bx]
        inc bx
        cmp al,0
        ;if null
        je bot
        int 21h
        jmp top
        bot:
        mov dl,13
        ;display CR + LF
        int 21h
        mov dl,10
        int 21h
    }
}
Using Data Structures

- Example 5 uses a data structure to store names, ages, and salaries.
- It then displays each of the entries by using a few assembly language procedures.
- The string procedure displays a character string, no carriage return/line feed combination is displayed—instead, a space is displayed.
- The Crlf procedure displays a carriage return/line feed combination.
  - the Numb procedure displays the integer
Example 5

// Program illustrating an assembly language procedure that
displays the contents of a C data structure.
// A simple data structure

typedef struct records
{
    char first_name[16];
    char last_name[16];
    int age;
    int salary;
} RECORD;

// Fill some records
RECORD record[4] =
{
    {“Bill”, “Boyd”, 56, 23000},
    {“Page”, “Turner”, 32, 34000},
    {“Bull”, “Dozer”, 39, 22000},
    {“Hy”, “Society”, 48, 62000}
};
// Program
void main(void)
{
    int pnt = -1;
    while (pnt++ < 3)
    {
        Str(record[pnt].last_name);
        Str(record[pnt].first_name);
        Numb(record[pnt].age);
        Numb(record[pnt].salary);
        Crlf();
    }
}

Str (char *string_adr[])
{
    _asm
    {
        mov bx,string_adr
        mov ah,2
        top:
        mov dl,[bx]
Example 5

```assembly
inc bx
cmp al,0
je bot
int 21h
jmp top
bot:
    mov al,20h
    int 21h
}
Crlf()
{
    _asm
    {
        mov ah,2
        mov dl,13
        int 21h
        mov dl,10
        int 21h
    }
}
Example 5

Numb (int temp) {
    _asm
    {
        mov ax,temp
        mov bx,10
        push bx
        L1: mov dx,0
        div bx
        push dx
        cmp ax,0
        jne L1
        L2: pop dx
        cmp dl,bl
        je L3
        mov ah,2
        add dl,30h
        int 21h
        jmp L2
        L3: mov dl,20h
        int 21h
    }
}
An Example of a Mixed-Language Program

• Example 6 shows how the program can do some operations in assembly language and some in C language. (Refer Example 7.6 from the book)

• The assembly language portions of the program are the Dispn procedure that displays an integer and the Readnum procedure, which reads an integer.

• The program makes no attempt to detect or correct errors.
THE END