CmpE213 – Digital Systems Design

Homework 8
ASM and C for the 8051

1. Enter the C-code outlined in problem 2 of homework 7 (the problem with a short lookup table) into Keil uVision. Create a project and add the code to it. Build the project and enter the debugger. Print out a copy of your dissassembled code (show both the C code and the ASM code using the "mixed" mode) while showing the function main(). Answer the following questions:

   (a) At what location did your "main" function actually begin in code space? What do you think all that extra code is for?

   (b) Did the compiler execute the instruction "P3=lookup[P1]" in the most efficient way possible? If not, show a more efficient way to do it in ASM.

   (c) Could the compiler have found a more efficient way to execute the instruction "for (i=0;i¡0x42;i++)"? If so, show a more efficient way to do it in ASM.

   (Note that if you had problems with problem 3 of the last homework, you can do the same sort of thing to see how the compiler executed that program).

2. Write a complete program in C which does the same basic thing as the following assembly code. You do not have to do it EXACTLY the same way (for example, I would not put references to the DPTR or ACC in your code). Note that this program doesn't really do anything important... it is here as an example. Test your code in Keil to make sure it works as you expect (however, Keil will probably compile your C code a little differently than it compiled below). Turn in a print-out of your C-code.

```
BSEG AT 22H
   x: DS 2
mydata segment data
rseg mydata
   y: DS 1
   table: DS 42
CSEG AT 0000H
   MOV x, #42H
   MOV x+1,#00H
   MOV A, #2AH
   CLR C
   RLC A
   MOV DPTR, #5280H
   MOVX @DPTR, A
   SETB 12H ; you may have to thing about this a second
```

```
MOV A, #table
ADD A, y
MOV R0,A
MOV @R0,x
INC R0
MOV @R0,x+1
```

3. Write a function called *delay()* which creates a delay simply by executing a loop which does nothing (e.g. *for (i=0;i<42;i++)*{;}). We'll see uses for this function in class soon. Give this function the following characteristics:

   - Pass the number of millisecond to wait (to be delayed) to the function as an unsigned char (for example, calling *delay(5)* would create a 5mS delay). The function returns nothing (void).
   - Create the delay using a do-while loop whose loop-variable is stored in <u>external</u> data memory.
   - VERIFY that passing a 1 to delay() creates a 1mS delay, a 2 creates a 2mS delay, etc, using the Keil debugger (+/- a few uS is OK). Finding the right number of iterations to perform the loop will take some experimentation. You may have to use two nested loops.
   - Define the number of loop-iterations needed to create a 1mS delay as a constant at the beginning of your code and use that constant in your function. That is, instead of using something like:

     do{ *blah* }*while(x* < 5280*);*

     use:

     #define DELAYCYCLES 5280
     do{ *blah* } *while(x <DELAYCYCLES);*.

     Using constants in this manner is good programming practice – it makes your code more understandable and makes it easier to change in the future.

Create and simulate your code using the Keil debugger to ensure it works and to find the correct number of loop iterations. Turn in a hardcopy of your code and email a copy of the file containing the function to daryl@ece.umr.edu. Please include the file as an attachment and use the subject header "HWK8 - program". **Please put your name at the top of your program (and anyone you worked with) and comment your code appropriately.**