CpE213 HW5 – Keil uVision IDE
Due Date: Thursday Oct. 23, 2003

1. Enter and compile the following program in uVision2 and then simulate and debug it.

```
      mystuff     segment     data
      rseg        mystuff
      blah:       DS          1
      mycode      segment     code
                  rseg  mycode
1                 MOV 80H, #0FFH    ;initialize P0 to 0FFH
2                 MOV A,#10
3                 MOV R0,#20H
4     LOOP: JNB P0.2,NEXT
5                 JMP LOOP
6     NEXT:       MOV @R0,80H
7                 CLR 07H
8     TIME: DEC A
9                 JNZ TIME
10                MOV mystuff,#42H
11                NOP
12                NOP
13                END
```

(a) Set the watch window to watch the value in the accumulator, in R0, and in variable blah.
(b) Single step through lines 1-5. Will the entire code segment be executed with the current values in P0? Why or why not? Alter the value of P0 *within the debugger* to enable the program to continue execution. Single step through lines 6 and 7. Display and record the contents of data memory location 20h, using the memory window.
(c) Find the opcode for instructions 1 and 2 by displaying the contents of code memory starting at c:0000 or by displaying the disassembly window. Note that the first instruction begins at c:0000 in code memory. Record the appropriate values for the opcode.
(d) What are the memory addresses of LOOP, NEXT, TIME, and mystuff?
(e) What type of jump did the assembler use to implement the "JMP LOOP" instruction (look at the dissassembled code)?
(f) Find the number of machine cycles and seconds it takes to execute the loop at instructions 8 and 9 in the following manner: Single step down to line 8 (you should already be there). Set a breakpoint on line 10. Record the cycles and seconds found on the register display. Execute (i.e. GO!) the loop at lines 8 and 9 until the breakpoint is reached. The time it took to execute the loop is the difference between the recorded cycles/times and the values now shown in the register display. Record this difference.
(g) If this code had worked like the programmer intended, the value 0FFH should have ended up in data memory location 20H. One instruction was written incorrectly... find that instruction and show what the correct instruction should be.

2. Write an assembly language program (complete with segment and variable declarations) to perform the following tasks (in the order given). Be sure to set aside memory for the stack and to initialize SP.

(a) Load external memory location 14A0H with the value 42H.
(b) Set the value of R4 to 42H in register bank 3 and set the register R6 to 2AH in register bank 1. Use the PSW and register addressing (instead of direct addressing). Verify-that the correct values have been set by observing either the register window or the memory window in uVision2.
(c) Set the value of R3 to 13D in register bank 2 and set the register RO to 01110101B in register bank 0 using direct addressing.
(d) Use the PUSH and POP operations to place several bytes onto an internal stack. Record the values in data memory and the locations affected, noting the final value of the stack pointer. What problems occur if you do not initialize the stack pointer? How might you avoid these problems?
(e) Place 42H into the accumulator if bit 2 of P1 is set (equals 1). Otherwise, place the value at external memory location 5280H into the accumulator. Perform this operation twice, once using byte-based instructions and once using bit-based instructions. When debugging this instruction sequence, be sure to test both the case where PI.2 is set and where P1.2 is cleared.

Simulate in uVision2 to prove that your program works. When you simulate, you want to watch the values in the SFRs, in data memory, and in xdata memory and want to watch the "flow of program execution" (i.e. which instructions are executed) to make sure your program does what you expect it to. Set up watch variables in uVision, and watch locations in memory. Form a good idea of what you expect your program to do. Step through your code one instruction at a time. If it does what you expect it to, then it probably works.

Turn in a) a printout of your code and b) a screen dump of uVision2 after your program completes.

The screen dump should be obtained in the following manner. Using the 'd' command, display the appropriate data memory locations to show the results of the operations performed in steps a-c. Make sure that the register contents are also displayed. Hit the shift key and "print screen" key at the same time. Open a word processor such as Microsoft Word and create a new document. Hit edit and paste to place the screen dump into the document. Now print the document.

3. The following code won't compile. What, do you think the problem is?

```
MOV A, #DOH
```

4. Is it generally better to declare segments like

```
DSEG
      myvar:  DS 1
```

or

```
mydataseg SEGMENT DATA
rseg nydataseg
myvar:  DS 1
```

Explain why.