CpE213 Assignment 7
Due at 3pm on Friday Oct. 21

1. For the following code.

```
start:  MOV 2AH,#5
        MOV R0,#80H
        CLR A
loop:   MOV @R0, A
        INC R0
        DJNZ 2AH,loop
stop:   SJMP stop
```

a) Find the final value of any registers or memory changed by executing this code segment.

b) Find the number of bytes in code memory this instruction sequence occupies.

c) Find the number of machine cycles, clock cycles and total amount of time this code takes to complete (assume it is complete when it executes SJMP stop for the first time). Assume the clock frequency is 12 MHz.

d) Explain why this code does not change the value of special function registers, such as P0, SP, DPL, and so on, located at internal memory locations 80H, 81H, and so on.

e) Write the machine code and corresponding code memory locations for the above code segment (use a table). Assume that code memory starts at 0000H

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-e. 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, #E0H
```

4. Is it generally better to declare segments like

```
DSEG
      myvar:  DS 1
```

or

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

Explain why.