CmpE213 – Digital Systems Design

Homework 6
ASM and C for the 8051

1. Briefly describe 4 possible projects you might do for CpE213. Describe any inputs or outputs to the system you would build and describe what it would do, for each project.
2. How many variables can be declared and storage allocated within the internal RAM of the 8051 for each of the following data types (assuming 128 bytes of internal RAM):
   (a) unsigned char
   (b) unsigned int
   (c) unsigned long
   (d) float
   (e) double
3. Write an ASM program to a) add two unsigned char variables, x and y, b) add two unsigned int variables, v and w, and c) write results to P1 and P2. This ASM program should be complete with segments, variables, labels, etc. In particular, give your program the following characteristics:
   - Declare x as a 1-byte variable in a relocatable segment of directly-addressable internal memory
   - Declare y as a 1-byte variable in an absolute segment of external memory beginning at 0000H.
   - Declare v as a 2-byte variable in an absolute segment of internal memory beginning at 70H.
   - Declare w as a 2-byte variable in a relocatable segment of internal indirectly-addressable memory (Hint: by defining w in indirectly addressable memory, you must access it using @R0 or @R1).
   - Write your instructions in an absolute code segment beginning at 0000H. In this code segment, perform the following operations:
     (a) Add x and y, putting the result in x.
     (b) Add v and w, putting the result in v.
     (c) Write x out to port 1.
     (d) To indicate to an external device that it should read the value of x from port 1, toggle bit 0 of P2 "on" for 5 machine cycles, then "off" for 5 machines cycles, then back on again (creating a low "RD-like" pulse).

Write and assemble your code in uVision and debug (simulate) in dScope. Be sure to double check that P2.0 is toggled at 5-Machine-Cycle intervals. Turn in a) a print-out of your code, b) a screen-dump of dScope after your program completes. Windows that should be displayed include: the module window, memory window showing internal direct memory, and the register window. **Please put your name at the top of your program (and anyone you may have worked with) and comment your code appropriately.**

Finally, email a copy of your ASM program (only) to daryl@ece.umr.edu. Please include your program as an attachment (not as ASCII text) and use the subject header "HWK5 - program".

4. Write a complete ASM program which performs the same function as the following C code. Assume the array, x, is in EXTERNAL DATA space, while all other variables are in internal data space.

```
unsigned int XDATA x[10]; // a 2-byte unsigned integer in external data space
unsigned char i,j;
main(){
  // read tank levels
  for (i=0;i<10;i++){
    P1 = i;  // select the tank to read
    j = 0;   // start a loop to do nothing (wait)
    do{
      j++;
    }while(j<0x2a)
    x[i] = P3;  // read the tank level and store for later use
  }
  while(1){;}
}
```

Compile your code and simulate it to be certain it works. Send me your ASM code (only) as an attachment to an email. Turn in a print-out of your code.

For those of you who might be interested in a purpose to this code, we might imagine that we have a 10-channel A/D (analog-to-digital) converter connected to port 1. This A/D converter is reading analog voltages from some sensor – say a set of 10 level sensors, which return an analog voltage proportional to the amount of water in 10 different tanks. The channel (tank from which a reading is taken) is selected using P1. If P1 is set to a binary 2, we're reading from tank 2 and so on. The A/D is read from P3. The loop simply a) selects a channel to read from, b) waits a bit for the A/D converter to produce a digital reading of the analog voltage, and c) writes that digital reading to the array x. This code represents just part of a larger program – a "real" program would have gone on to do something with these tank levels.