Name: _____
NetID: _____@wisc.edu

Homework 5

CS/ECE 252 Section-2 (MWF 11:00)

Assigned on October 14th
Due on Friday, October 23rd by the beginning of class (11 AM)
Submit by hard copy unless otherwise stated to submit via the browser-based simulator. Neat and legible handwriting is preferred, especially for your name and NetID.

1. Convert the following 32 bit floating point hexadecimal number to its decimal value (as per the scheme mentioned in the book): 3EA00000

Hint: Note that this conversion cannot be performed directly. Hexadecimal numbers must first be converted into intermediate binary in order to do the floating point conversion. Also, this is a 32 bit floating point number. Refer to section 4.5 in the textbook for the scheme of the bits (1 bit for the sign, 8 bits for the exponent, and the remaining 23 bits for the fraction).

Show your work for each step of the conversion for full credit. (2)

2. There are three parts of an ISA. What are they? Explain each in **detail**. **(3)**

3. For the following types of memory on the AVR chip, describe their **type** (persistent or non-persistent), **size** (depth and width), **speed**, and **function**(s). **(3)**

   (a) program memory
   type:
   size:
   speed:
   function:

   (b) RAM
   type:
   size:
   speed:
   function:

   (c) SREG (status register)
   type:
   size:
   speed:
   function:

4. What is the difference between ISA and assembly language? **(1)**

5. What is non-persistent memory and why is it useful? **(1)**

6. What does an assembler do? **(1)**

7. What registers can be used in the ldi instruction? **(1)**

8. There are 7 control flow instructions that alter the program counter by more than 1: **breq, brne, brsh, brlo, rjmp, rcall,** and **ret**. We can roughly categorize these as branching, jumps, and return instructions. When would you use one type of instruction over the other? **(2)**

9. For the following assembly code, trace through it by hand and write the N, C and Z flag and PC values after each line for the first 8 lines executed. Assume the initial values of all registers to be 0. Note that AVR effectively uses an 8-bit representation, and the effect of this fact on the N-flag. Check your answer with the browser-based AVR simulator. **(2)**

Instruction	**N**	**C**	**Z**	**PC**
`ldi r16,100`	0	0	0	1
`ldi r17,244`				
`add r17,r16`				
`add r17,r17`				
`sub r16,r17`				
`ldi r18, 1`				
`cp r16, r18`				
`breq -3`				
`ldi r19, 0`				

10. Consider the code snippet below.

(a) Trace through the code by hand and fill out the table below. Assume all registers are initialized at 0. Check your answer with the browser-based AVR simulator. **(2)**

```
ldi r16, 10     ; line i
ldi r17, 8      ; line ii
inc r17         ; line iii
cp r16, r17     ; line iv
brne -3         ; line v
halt            ; line vi
```

Current line	**Calculation performed by this line**	**Registers**		**Next line:**
		r16	**r17**	
i	load 10 into r16	10	0	ii

(b) Hopefully, you were able to fill out the table and not run into an infinite loop. Will you get an infinite loop if the second line of the program was “ldi r17, 11” instead? If yes, what makes it infinite? If no, at what point will the loop stop? (Hint: what happens if a register overflows?) **(1)**

11. Write assembly code to configure the **complete** PORT D as an output port and write 134 on this port. **Annotate your code with comments for full credit. Submit using the browser-based simulator. (2)**

12. Using the stack, Write a simple **assembly language** function labeled *myfunction* that compares two values ~~in the top two entries of the stack~~ and prints the higher number to the output LED. If they are equal, print either value.

    Initialize your stack to RAM location 3000. ~~Push values 4 and 3 onto the stack and then call this function.~~ Run your function with the values 3 and 4. **Annotate your code with comments for full credit. Submit using the browser-based simulator. (4)**

13. Convert the following Python code into **assembly language**. Assume x is stored in r16, y in r17, and z in r18, and a in r19. You may store values in other registers if necessary. **Annotate your code with comments for full credit. Submit using the browser-based simulator. (4)**

```
x = 2
y = 1
z = x + y
if ( z >= 2):
     a = 1
else:
     a = 2
```

14. Repeat Homework 2 Question 11, but this time use **assembly language** instead of Python. **(6)**

    A hard coded solution will result in 0 points. **Annotate your code with comments for full credit. Submit using the browser-based simulator.**

    For reference, Homework 2 Question 11:

    > Write a program to print the first n Fibonacci numbers. (Initialize n as 15). Start with 0 and 1 as the first two numbers. The next number is created by adding the previous two numbers. Thus, the series would go like this: 0 1 1 2 3 5 8 13 21 34 55 89 144 233 377.

~~*Note:* Due to overflow, it is okay for your program to instead print 0 1 1 2 3 5 8 13 21 34 55 89 16 105 121~~

Note: Due to overflow, it is okay for your program to instead print 0 1 1 2 3 5 8 13 21 34 55 89 144 233 **121**

For reference, answer to Homework 2 Question 11:

```
n = 15
counter = 1
a = 0
b = 1
print(a)
while(counter<n):
    print(b)
    temp = b
    b = b + a
    a = temp
    counter = counter + 1
```

For reference, answer to Homework 2 Question 11 with a hard coded solution:

```
print(0)
print(1)
print(1)
print(2)
print(3)
print(5)
print(8)
print(13)
print(21)
print(34)
print(55)
print(89)
print(144)
print(233)
print(377)
```

For reference, answer to this question with a hard coded solution:

```
; this will earn you 0 points because it is hard coded
; this is provided to give you a sense of what we are
;    looking for and how the Output LCD should behave
;    when we run your code
; this also demonstrates how to print to Output LCD
ldi r16, 255 ; set r16 to all high bits
out 17, r16  ; all high bits IO 17 for output
ldi r17, 0   ; 1st number
out 18, r17  ; print 1st number
ldi r17, 1   ; 2nd number
out 18, r17  ; print 2nd number
ldi r17, 1   ; 3rd number
out 18, r17  ; print 3rd number
ldi r17, 2   ; 4th number
out 18, r17  ; print 4th number
ldi r17, 3   ; 5th number
out 18, r17  ; print 5th number
ldi r17, 5   ; 6th number
out 18, r17  ; print 6th number
ldi r17, 8   ; 7th number
out 18, r17  ; print 7th number
ldi r17, 13  ; 8th number
out 18, r17  ; print 8th number
ldi r17, 21  ; 9th number
out 18, r17  ; print 9th number
ldi r17, 34  ; 10th number
out 18, r17  ; print 10th number
ldi r17, 55  ; 11th number
out 18, r17  ; print 11th number
ldi r17, 89  ; 12th number
out 18, r17  ; print 12th number
ldi r17, 144 ; 13th number
out 18, r17  ; print 13th number
ldi r17, 233 ; 14th number
out 18, r17  ; print 14th number
ldi r17, 121 ; 15th number, wrap around, 377-256=121
out 18, r17  ; print 15th number
halt         ; end program
```

15. Repeat Homework 2 Question 12, but this time use **assembly language** instead of Python, and go from 32 to 50 instead. **(6)**

    Annotate your code with comments for full credit.

    Use the browser-based simulator to write and submit.

    Hints:

    You may have to write your own multiply by doing multiple adds.

    You may have to write your own divide by using a quotient counter, and subtracting the divisor from the dividend while the dividend is greater than the divisor.

For reference, Homework 2 Question 12:

Say you wanted to print out the Celsius equivalent for all integer Fahrenheit temperatures from **32** degrees F to 50 degrees F. Write a program to print out this conversion information. The pseudocode for implementing this is given below.

The equation for converting Fahrenheit (F) to Celsius (C) is: $C = (F - 32) * \frac{5}{9}$

i. Set F's initial value to **32** (lower bound)
ii. While F is less than or equal to 50 (upper bound)
iii. Convert Fahrenheit to Celsius.
iv. **Print the number of degrees in Fahrenheit**
v. **Print the number of degrees in Celsius**
vi. Increment F

For reference, answer to Homework 2 Question 12:

```
F = 32
while(F <= 50):
    C = (F - 32) * 5 / 9
    print(F)
    print(C)
    F = F + 1
```

Sample output:

```
32
0
33
0
34
1
35
1
36
2
37
2
38
3
39
3
40
4
41
5
42
5
43
6
44
6
45
7
46
7
47
8
48
8
49
9
50
10
```