Name: _____

Write all of your responses on the extra paper provided. Hand in this exam paper along with your solutions, please place your name on the top of each page. Show all of your work.

1 ASC: A Simple Computer Exercises

Each question is worth 20 points.

1. For each of the following load functions, give the effective address of the memory location being addressed and the contents of the accumulator resulting from the command. Assume that the contents of memory and the index registers are as follows and that all indirect indexing mode addresses are preindexed. Also, the symbol Z has value C and Y has value 11, both in hexadecimal. Memory and register contents are also in hexadecimal.

Address	Contents
0	4
1	E
2	3
3	12
4	A
5	2
6	5
7	10
8	C
9	A
A	3
B	5
C	7
D	1
E	0
F	0
10	0
11	AA
12	1D
13	3
14	5
15	C1
16	D
17	8
18	2
19	F
1A	9
1B	E

Index Register	Contents
1	A
2	7
3	4

(a) `LDA Z`

Address	Accumulator

(b) `LDA* Z`

Address	Accumulator

(c) `LDA Y, 2`

Address	Accumulator

(d) `LDA* Z, 1`

Address	Accumulator

(e) `LDA 8, 1`

Address	Accumulator

(f) `LDA* 17`

Address	Accumulator

(g) `LDA* 17, 3`

Address	Accumulator

2. Trace through the following ASC program with the following inputs and give the outputs of the program. Assume that the program is named prog.

```
        ORG  0
START   RWD
        STA  Z
        LDX  Z, 1
        RWD
        STA  Z
        LDX  Z, 2
        LDX  =2, 3
        LDA  X, 2
        STA  Z
        LDA  =5
LOOP    STA  Y
        ADD  X, 1
        WWD
        TIX  NEXT, 1
NEXT    TDX  LOOP, 2
        LDA  Y
        WWD
        LDA  Z
        WWD
        HLT
X       BSC  1A, 2, B, C1, 4, A, 3
Y       BSS  1
Z       BSS  1
        END  START
```

   (a) `./prog 1 3`

   (b) `./prog 2 4`

3. Construct the Symbol Table for the ASC program in the previous exercise. Then assemble the code in both binary and hexadecimal form. The ASC opcode table is below.

Mn.	Op.
HLT	0
LDA	1
STA	2
ADD	3

Mn.	Op.
TCA	4
BRU	5
BIP	6
BIN	7

Mn.	Op.
RWD	8
WWD	9
SHL	A
SHR	B

Mn.	Op.
LDX	C
STX	D
TIX	E
TDX	F

2 NASM Coding

4. (*25 points*) Write the following program using the NASM assembly language. You may use the atoi, itoa, iprint, iprintLF, sprint, sprintLF, slen, and quit subroutines that we constructed in class, remember to add the `%include 'functions.asm'` at the beginning.

   Fibonacci Number Program: The Fibonacci sequence starts with two ones and then every number after that is the sum of the two previous entries in the sequence. So the sequence is: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ... We define Fib(n) to be the n^{th} Fibonacci number, specifically, Fib(1) = 1, Fib(2) = 1, Fib(3) = 2, Fib(4) = 3, Fib(5) = 5, Fib(6) = 8, and so on. Write a program in NASM that computes the n^{th} Fibonacci number. Hard code the value of n into one of the registers so if the programmer would change that single value the new Fibonacci number would be calculated. This should be the only line that a programmer should have to change to get the altered result.

5. (*25 points*) Do one and only one of the following programs. Write the program using the NASM assembly language. You may use the atoi, itoa, iprint, iprintLF, sprint, sprintLF, slen, and quit subroutines that we constructed in class, remember to add the `%include 'functions.asm'` at the beginning.

   (a) Factorial Program: A simple loop based program to calculate the factorial of a number. Recall that $n! = n \cdot (n-1) \cdot (n-1) \cdots 2 \cdot 1$, if n is greater than 0, and we define $0! = 1$. Hard code the value of n into one of the registers so if the programmer would change that single value the new factorial value would be calculated. This should be the only line that a programmer should have to change to get the altered result.

   (b) Factoring Program: A brute-force factoring program that will find the first prime factor of a number n. Hard code the value of n (the number to be factored) into one of the registers so that if a programmer would change that single value the altered factorization would result. This should be the only line that a programmer should have to change to get the altered result.

   (c) Greatest Common Divisor Program: Using the Euclidean Algorithm one can find the Greatest Common Divisor of two numbers, without factoring them. The algorithm is fairly simple, first write $a = b \cdot q_1 + r_1$ where q_1 and r_1 are the quotient and remainder, respectively, of $\frac{a}{b}$. Then shift b to the other side of the equation, shift the first remainder r_1 to the position of b and repeat the process. Then shift and repeat, continue until we get a remainder of 0. When that happens, our answer is the previous remainder.

$$\begin{aligned} a &= b \cdot q_1 + r_1 \\ b &= r_1 \cdot q_2 + r_2 \\ r_1 &= r_2 \cdot q_3 + r_3 \\ &\vdots \\ r_t &= r_{t+1} \cdot q_{t+2} + 0 \end{aligned}$$

So $\gcd(a, b) = r_{t+1}$.

Hard code the values of a and b (the numbers to be gcded) into two of the registers. If a programmer would change those numbers, reassemble and link the program, then the result would be the GCD of the altered numbers.