1. Given the binary bit string **1 0 1 0 1 1 0**

   What decimal number does it represent if it is stored in:

   (1) Unsigned format?
   (1) Sign-Magnitude format?
   (1) Excess-64 format?
   (1) Two's complement format?

   (1) What **character** does it represent?

2. Given the following Base 4 number: **1 3 0 2**

   (1) What is the Binary representation of this number?
   (1) What is the Octal representation of this number?
   (1) What is the Decimal representation of this number?
   (1) What is the Hex representation of this number?

3. (6) Assuming a floating point format where the most significant bit of the word is the sign bit for the mantissa, the next 5 bits are the exponent stored in Excess-N format, and the remaining 10 bits are the magnitude of the mantissa, write down the bit pattern for the following decimal number (Assume the format does use hidden one's):

   -1.375×10^{1}

4. (10) Describe how immediate, direct, register direct, register indirect and indexed addressing modes work. (Pictures are **preferred**.)

5. (4) Describe the 3 step sequence that occurs continuously while a computer is turned on.

6. (2) Are the PC and XR registers the same size? Why or why not?

7. (2) What is the maximum amount of memory (in bytes) that CUSP can address?

8. (2) How *does* the CPU distinguish between instructions and data?

9. (15) Given the following code sequence:

Program **Show your work area**

```
        .PAGE
        .EQU    @,$000
        .WORD   8
        .BLKW   3,$FFF
        .EQU    A,$00B
        LDX#    $002
        LDA     $008
        ADA#    $003
        STA+    $006
        .EQU    @,$008
        LDA#    $013
        CMA     $00F
        JGT     DONE
        DEC     VAR1
        TAX
DONE:   HLT
VAR1:   .WORD $040550
        .WORD $000015
        .WORD $13A001
        .END
```

Symbol Table	

Fill out the symbol table. What memory locations and registers are **altered** by the *execution* of this program? Show the values of the registers and memory locations that have **changed** when the program terminates. Assume the PC is incremented before the current instruction is executed. (Fetch, Decode, Inc PC, Execute).

ACC = **XR =** **PC =** **SP =**

MEM[ ] = **MEM[ ] =** **MEM[ ] =** **MEM[ ] =**

MEM[ ] = **MEM[ ] =** **MEM[ ] =** **MEM[ ] =**

10. (20) When executed, do the following code fragments produce the same end result in the ACC and memory? If not, where do they differ? (Show your work, step by step).

Code Fragment 1

```
 .EQU          @,$100
LDA            $109
LDX#           1
ADA+           $109
ADX#           2
SBA            $109
DIV+           $109
HLT
 .BLKW 3, $FFE
```

Code Fragment 2

```
 .EQU          @,$100
LDX#           3
LDA+           Label+8
ADX#           3
ADA+           Label+6
SBX#           2
SBA            Label+11
DIV+           Label+11
HLT
 .EQU          Label, $0FE
 .BLKW 3, $FFE
 .BLKW 4, $BAD
```

11. (25) Your job, should you decide to accept it, is to assemble the following program and create a list file output. You should also fill out the symbol table as you go. (Just create the list file for the first 12 lines). You don't need to fill out all the source lines, just enough so I know which line you are on.

Assembly Language Program:

```
        .EQU    @,$040
        .EQU    J,129
K:      .BLKW   2,1
L:      .WORD   $F02B36
        .EQU    M,1
N:      .WORD   2
        .EQU    P,7
        LDX#    M
        LDA+    K
        STA     N
        HLT
        .END
```

Symbol Table	

LINE	ADDR	CONTENTS	SOURCE LINE
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			

ASCII Table (MSD = Most Significant Digit)																
MSD (Hex)	**Least Significant Digit (Hex)**															
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
2	space	!	"	#	$	%	&	,	(	)	*	+	’	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	ˆ	_
6	‘	a	b	c	d	e	f	g	h	i	h	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	˜	del

Instructions:

Instruction	Opcode (in Hex)
LDA	00
LDX	01
LDS	02
LDF	03
STA	04
STX	05
STS	06
STF	07
ADA	10
ADX	11
ADS	12
ADF	13
SBA	14
SBX	15
SBS	16
SBF	17
CMA	20
CMX	21
CMS	22
CMF	23
HLT	FFFFFF

Addressing Modes:

Mode	Opcode
Immediate	0
Direct	2
Indexed	4