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

   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: **3 2 1 0**

   (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 12-bit 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 6 bits are the mantissa, write down the bit pattern for the following decimal number (Assume the format does use hidden one's):

   $-.875 \times 10^{1}$

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

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

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

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

8. Suppose you have a machine with the following characteristics:

The left most 5 bits store the opcode
the next 4 bits tell how to find the Destination/Source1 operand
the last 4 next to it hold the Source2 operand

a.(1) Draw a picture of the instruction. How many bits does it have in total?

b (2). What is the maximum number of different opcodes (instructions) this machine could support?

c.(2) Assuming the machine only supports memory direct addressing, how many different memory locations can it access?

d.(3) Assuming the machine supports both memory direct and register direct addressing, and the instruction size does not change (still 5 bits in the opcode field, etc.), how many different registers can be accessed?

e.(2) Assuming the registers are the same size as the instructions, how many memory locations can be accessed if the machine supports register indirect addressing?

9. 9) Explain the effects of the following instruction sequence on the registers, flags, and memory locations that are indicated. Assume that all programs start at memory location $000, and that the PC is incremented before execution of the instruction. If a value cannot be determined from the information in the instruction fragment, then indicate so by using ???

```
LDA     $004
TAX
JLT     $004
LDA#    $000
HLT
```

PC =
ACC =
XR =
LT =

10. (28) Given the following code sequence:

Program		Show your work area
	.PAGE	
	.EQU	@,$10
ARR:	.BLKW	3,$009
TMP:	.WORD	$B
	.EQU	BOB,$00E
BEGIN:	LDX	TMP
	LDA#	ARR
	ADA	SALLY+1
	STA+	BOB
	.EQU	@,$18
	LDA	$020
	CMA	SALLY
	JGE	DONE
	DEC	SALLY
	TAX	
DONE:	HLT	
SALLY:	.WORD	$040555
	.WORD	$202010
	.WORD	$202020
	.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), and that the execution begins at "BEGIN". Show your work if you want any chance of partial credit ...

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

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

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

11. (20) 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    @,$200
        .EQU    A,2
        .EQU    B,63
C:      .WORD   $F02B36
D:      .WORD   9
        LDA#    C
        .EQU    E,100
        STA+    A
        LDX     D
        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	**Least Significant Digit (Hex)**															
(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