1. (6) Saving register values before making a procedure call can be done according to one of two conventions. Briefly describe the two approaches, and give an advantage for each.

2. (6) Many changes had to be made to CUSP in order to support interrupts. Describe 6 of them.

3. (4) In CUSP, the SP points to a certain item and grows in a certain way. In class, we talked about 3 other ways to manage a stack. Describe all 4 methods (include CUSP).

4. (2) CUSP uses a Frame Pointer. Why? What is it used for?

5. (2) Suppose the following code is executed:

```
LDA#    $E0
OUTB    $30
INB     $30
HLT
```

What does the Accumulator contain after the HLT instruction is executed?

6. (12) Given the following code sequence:

```
        .EQU    @,$000
V1:     .WORD   $2
V2:     .WORD   $3
V3:     .WORD   $4
START:  LDX#    V2
        LDC     START+1
        STC     V3+5
        LDX     V3
        STC*    V1
        HLT
        .END
```

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 program begins execution at "START".

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

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

6. (30) Given the following assembly language program, fill out the symbol table, and write down the contents of the registers after the execution of each instruction. (Assume you have begun single-stepping with the instruction at location $106).

Assembly Language Program:

```
            .EQU    @,$100
V0:         .WORD   $FA4102
V1:         .WORD   $102
            .EQU    V2, $ABC
V3:         .WORD   $2EA047
V5:         .WORD   $101
            .WORD   $100
            .WORD   $8D10FD

BEGIN:      LDA     V0
            LDA+    BEGIN-5
            LDA#    V3
            LDA&    V1
            POPA
            LDC     V1
            LDC*    ! $005
            LDA&    ! $004
            LDS+    ! $003
            LDX*    V5+1
            .END
```

Symbol Table	

Before LDA Inst.	**ACC=$3**	**SP=$102**	**XR=$3**	**FP=$0FF**	**PC=$106**
After LDA Inst.	**ACC=**	**SP=**	**XR=**	**FP=**	**PC=**
After LDA+ Inst.	**ACC=**	**SP=**	**XR=**	**FP=**	**PC=**
After LDA# Inst.	**ACC=**	**SP=**	**XR=**	**FP=**	**PC=**
After LDA& Inst.	**ACC=**	**SP=**	**XR=**	**FP=**	**PC=**
After POPA Inst.	**ACC=**	**SP=**	**XR=**	**FP=**	**PC=**
After LDC Inst.	**ACC=**	**SP=**	**XR=**	**FP=**	**PC=**
After LDC* Inst.	**ACC=**	**SP=**	**XR=**	**FP=**	**PC=**
After LDA& Inst.	**ACC=**	**SP=**	**XR=**	**FP=**	**PC=**
After LDS+ Inst.	**ACC=**	**SP=**	**XR=**	**FP=**	**PC=**
After LDX* Inst.	**ACC=**	**SP=**	**XR=**	**FP=**	**PC=**

8. (20) Here is the beginning of some code that calls a subroutine, and the first few statements of the subroutine itself. Write down the contents of the Stack at the point where the subroutine stops. If you do not know the exact contents, write down what they would be ("contents of Accumulator", for example). Assume execution begins at "START" (location 00A).

Assembly Language Program:

```
            .EQU    @, 000
            LDS#    $E00
            TSF
            LDX#    9
            LDA#    3
ARRAY:      .BLKW   4,7
COUNT:      .WORD   0
LNTH:       .WORD   5

START:      PSH#    COUNT
            PSH#    ARRAY
            PSH     LNTH
            JSR PARRAY
            HLT

            .EQU    @,$100
            .EQU    CNT,4
            .EQU    ARR,3
            .EQU    LENTH,2

PARRAY:     BGN#    2
            PSHA
            PSHX
```

(*Write down Contents of Stack at this point, as well as the value of the Stack Pointer and the Frame Pointer*)

Mem Address	**Mem Contents**
SP=	**FP=**

9. (18) The following is an interrupt-driven print string program. This program will not work as shown - there are at least 3 lines missing that are necessary in order to make the program function correctly, and 3 others that are incorrect. Your task is to fill in the missing lines and fix the incorrect ones.

```
                .EQU        @,$000
MAIN:           LDA#        $80
                OUTB        $10         ; Enable interrupts
LOOP:           INB         $10         ; get status of printer
                JGE         LOOP        ; check to see if it is ready
                LDX#        1
                LDC         STRING
                OUTB        $11         ; send character to printer
                ADX#        1           ;
                STX         INDEX       ; update index
                LDS#        $E00
CYCLE:          NOP
                JMP         CYCLE
ISR:            PSHA                    ; save registers
                PSHX
                LDX         INDEX       ; Pick up string char.
                CMX#        LENGTH      ; Check to see if we are done
                JLT         DONE
                LDC         STRING      ; output not complete -
                OUTB        $11         ; print character
                ADX#        1
                STX         INDEX       ; store updated index
                JMP         EXIT
DONE:           LDA#        $C0         ; output complete -
                OUTB        $10         ; clear interrupt pending bit
EXIT:           POPX                    ; restore regs
                POPA
                RTN
INDEX:          .WORD
STRING:         .CHAR       'Print this!',LENGTH
```

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	[	\	]	^	_
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
LDC	50
STA	04
STX	05
STS	06
STF	07
STC	51
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
Frame Immediate	1
Direct	2
Frame Direct	3
Indexed	4
Frame Indexed	5
Indirect	6
Indirect Indexed	8

I/O Port Information:

I/O Port	Register
$000	Keyboard Control
$000	Keyboard Status
$010	Printer Control
$010	Printer Status
$020	Tape Drive Control
$020	Tape Drive Status
$030	Timer Control
$030	Timer Status
$316	CRT Control

Keyboard		
Register	Bit Number 7654 3210	Interpretation
Control	x--- ---- -x-- ---- --xx xxxx	1 = enable interrupts, 0 = disable interrupts 1 = flush buffer, 0 = no operation unused (no affect)
Status	x--- ---- -x-- ---- --xx xxxx	1 = ready (data available) 1 = interrupt enabled unused (always zero)
Interrupt Addr		$FF8

Tape Drive		
Register	Bit Number 7654 3210	Interpretation
Control	x--- ---- -x-- ---- --xx ---- ---- xxxx	1 = enable interrupts, 0 = disable interrupts 1 = clear interrupt request, 0 = no operation 00 = no operation 01 = read record 10 = write record 11 = rewind tape unused (no affect)
Status	x--- ---- -x-- ---- --x- ---- ---x ---- ---- 1--- ---- -xxx	1 = ready (to begin new operation) 1 = interrupt enabled 1 = tape mounted 1 = interrupt pending 1 = end of tape encoutered on read unused (always zero)
Interrupt Addr		$FFA

Printer		
Register	Bit Number 7654 3210	Interpretation
Control	x--- ---- -x-- ---- --xx xxxx	1 = enable interrupts, 0 = disable interrupts 1 = clear interrupt request, 0 = no operation unused (no affect)
Status	x--- ---- -x-- ---- --x- ---- ---x ---- ---- xxxx	1 = ready (to receive character) 1 = interrupt enabled 1 = printer on-line 1 = interrupt pending unused (always zero)
Interrupt Addr		$FF9

Timer		
Register	Bit Number 7654 3210	Interpretation
Control	x--- ---- -x-- ---- --xx ---- ---- xxxx	1 = enable interrupts, 0 = disable interrupts 1 = clear ready bit, 0 = no operation 00 = no operation 01 = start timer (after loading counter) 10 = stop timer 11 = start timer (without loading counter) unused (no affect)
Status	x--- ---- -x-- ---- --xx xxxx	1 = ready (count complete) 1 = interrupt enabled unused (always zero)
Interrupt Addr		$FFB