ECE291 Exam I

October 2, 2001

200 Points

For this exam, you are permitted to use one double-sided sheet of 8.5" x 11" notes. No textbooks, calculators, computers, communication devices, or other notes are allowed. Read questions carefully. Giving or receiving aid on an exam is strictly forbidden. You will have 80 minutes to complete this exam. Good luck.

If you write more than a couple of sentences on any short answer question, you're probably giving too much information. Be concise but complete. You can use the back side of the paper if you need more space.

Name:		
Problem	**Points**	**Max**
1. Basics		30
2. Memory		30
3. Program Structure		30
4. Branching and Procedures		30
5. Coding		80
Total		200

1. Basics

A. Write **legal** or **illegal** by each of the following instructions. If an instruction is illegal, explain why. No credit will be given for illegal answers without explanation. 20 points.

Instruction	Legal or Illegal?
MOV [SI], [DI]	Illegal. Can't move memory to memory with a MOV instruction.
POP word [1234]	Legal.
ADD AX, BL	Illegal. Operands must be same size.
MOV CL, [BL]	Illegal. BL cannot address memory.
ADD [BX], 24	Illegal. Ambiguous data size.
MOV AX, [BX+SI+0ABh]	Legal.
MOV CS, [AX]	Illegal. AX cannot address memory.
MOV AX, [BX+BP]	Illegal. Can't add two base registers for memory offset.
MOV DS, CS	Illegal. Can't move a segment register directly to another segment register.
SUB 24, byte [BX]	Illegal. Destination cannot be immediate data.

B. Describe the difference between SUB and SBB. 3 points

SUB A, B → A = A – B
SBB A, B → A = A – B – CF

C. What information is kept in the first 1kb of memory? 3 points.

The Interrupt Vector Table.

D. Given the set of instructions:

```
MOV AL, 07Eh
ADD AL, 2
```

What will be the value of the requested flags after execution? 4 points.

Carry = 0

Overflow = 1

Sign = 1

Zero = 0

2. Memory

A. Using the register values listed below, determine the linear memory address (or addresses) of the byte or bytes read or written in each of the instructions.

Note that each problem can be solved independently
The first solution is given as an example
All values provided are in hexadecimal
Each instruction is worth 3 points for a total of 15 points

Register	Value
BX	1212
BP	AA33
ES	2000

Register	Value
DI	8000
SI	0005
SS	5400

Register	Value
CS	4400
DS	2300
SP	0555

Instruction	Linear memory address(es) written or read (hex)
MOV AX, [SI]	23005, 23006
SUB [BX+SI], CL	24217
PUSH SI	54553, 54554
AND AX, [BP+10h]	5EA43, 5EA44
OR DL, [ES:DI]	28000
CMP AL, [DI]	2C000

B. Write out the contents of memory that result from the following variable declarations beginning from offset 0 in the code segment. You may use 'H' to represent the ASCII byte for the character H, for example. If the contents cannot be determined then put an X in the box. 15 points.

```
SEGMENT CODE
        X       DB      7Fh, 80h
        Y       DW      0ABCDh
        Z       RESD    1
        ARR     DW      12h, 34h, 56h, 78h
        STR     DB      "HELLO",0
```

CS:00	7F	80	CD	AB	00 (X)	00 (X)	00 (X)	00 (X)
CS:08	12	00	34	00	56	00	78	00
CS:10	'H'	'E'	'L'	'L'	'O'	00	X	X

3. Program Structure and Debugging

A. Given the following procedure, write down as many syntax or structure errors as you can find. You will receive 3 points per error up to a maximum of 15 points. Note that an error that involves doing something in the wrong order, thereby affecting multiple lines of code doesn't qualify as multiple errors—just a single error. 15 points.

```
;assume C calling convention:   far 1LameProc(unsigned a, unsigned b, unsigned c)
;procedure calculates C*(A+B)
;return value in DX:AX

1LameProc:                                  1
                                            2
        setup the stack frame               3
        push bp                             4
        mov sp, bp                          5
                                            6
        ;preserve registers that get modified  7
        push ax                             8
        push bx                             9
        push cx                            10
                                           11
        mov ax, BP+Ah                      12
        add ax, [BP+8]                     13
        mul ax, [BP+6]                     14
                                           16
        ;restore register that got modified 17
        pop ax                             18
        pop bx                             19
        pop cx                             20
                                           21
        ret                                22
```

Line 1: Label should be prefixed with _
Line 3: Comment missing ;
Line 5: sp and bp are backwards
Line 8 and 18: Shouldn't save and restore AX
Line 9, 10, 19, 20: Popping in wrong order
Line 12: Need brackets around BP+Ah
Line 12: Change Ah to 0Ah
Line 12, 14: Referencing wrong data, 0Ah and 6 should be swapped
Line 14: Invalid multiplication syntax
Line 22: ret should be retf

B. Why is it important to keep ISR's as short as possible? 5 points.

So they don't block ISR's of lesser priority for extended periods of time.

C. How does the INT instruction differ from the CALL instruction? 5 points.

INT pushes the flag register in addition to the return address.

D. What is the stack frame? What is its purpose? 5 points

Creates a fixed reference point which a procedure can use to access arguments passed on the stack or create temporary variables.

4. Branching and Procedures

A. Given the following register values, determine the result of each provided jump instruction scenarios. Assume the instructions execute independently. In other words, use the provided register values for each set of instructions. (10 points)

AX = 1013 BX = A083 CX = FFFF DX = 0013

```
CMP   CX, DX
JNB   SomeLabel          Jump

CMP   CX, DX
JG    SomeLabel          No Jump

INC   CX
JZ    SomeLabel          Jump

SUB   BX, BX
JE    SomeLabel          Jump

CMP   AL, DL
JNG   SomeLabel          Jump
```

B. How does JL differ from JB? 5 points

JL compares 2's compliment signed numbers.
JB compares unsigned numbers.

C. What events take place when a FAR CALL is executed? 5 points.

PUSH IP
PUSH CS
PUSHF

(not necessarily in this order)

D. Given the following memory dump and program fragments, provide the final value of AX for each program fragment. 10 points.

Note that each program fragment can be solved independently
All values provided are in hexadecimal

DS:00	12	34	56	78	9A	BC	DE	FF
DS:08	83	29	48	8A	91	BF	CE	90
DS:10	28	91	37	28	91	DB	BA	AC
DS:18	94	95	96	92	93	90	92	95
DS:20	08	09	0A	0B	0C	0F	0E	01
DS:28	02	03	04	05	06	07	11	12
DS:30	12	13	14	15	16	82	91	38
DS:38	AB	BC	CD	DE	EF	FA	AB	AC

Instruction(s)	Value of AX after execution of instruction(s)
MOV AX, [22h]	0B0Ah
MOV AL, [19h] MOV AH, [3Eh]	0AB95h

MYARRAY EQU 10h
MOV SI, MYARRAY
MOV AX, [SI+10h]
	0908

INC WORD[1h] MOV AX, WORD[0h]	3512

5. Coding

A. Write an assembly procedure that calculates the area of a rhombus. The formula is given as $A = \frac{1}{2}(B_1 + B_2)h$. The unsigned arguments are located in three word sized memory variables. The result should be returned in a fourth word sized memory location. Your procedure should preserve preexisting register values. (40 points)

```
B1          RESW        ; input B1
B2          RESW        ; input B2
H           RESW        ; input h
RESULT      RESW        ; result

AreaRhombus:
        ;Write your code here

        push    ax
        push    dx

        mov     ax, [B1]                    5 points
        add     ax, [B2]                    5 points
        mov     dx, [H]                     5 points
        mul     dx                          5 points
        ; multiply before divide            5 points
        shr     ax, 1                       5 points
        mov     [RESULT], ax                5 points

        pop     dx                          5 points
        pop     ax

ret
```

B. Write a procedure that converts all the '?' characters in a string to '!' characters. The starting address of the string is passed to your procedure on the stack. Your procedure should count the number of '?' found and modified and return that count value in the CX register. (40 points)

ChangeQtoBang:

```
Mov     bp, sp                  5 points (uses bp to access stack)

Mov     bx, [bp + 2]            5 points (loading init values)
Xor     cx, cx

.Loop
mov     al, [bx]                5 points (reads letters out correctly)

cmp     al, '$'                 5 points (terminates loop correctly)
je      .Done

cmp     al, '?'                 5 points (locates ?'s correctly)
jne     .continue

mov     byte [bx], '!'          5 points (changes ? to !)

inc     cx                      5 points (updates count)

.continue
inc     bx                      5 points (updates loop variables)
jmp     .Loop                           (and continues loop)

.Done
```

ret