

**ECE291 Exam II**  
**Tuesday April 3, 2001**  
*150 Points*

*For this exam, you are permitted to use two double-sided sheets 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 75 minutes to complete this exam. Good luck.*

*Unless otherwise specified, assume memory and register dumps are provided in hex.*

<b>Name:</b>		
<b>Problem</b>	<b>Points</b>	<b>Max</b>
1. Fundamentals		15
2. Interrupts		40
3. Video Graphics		30
4. MMX		30
5. Floating-Point Numbers		35
<b>Total</b>		<b>150</b>

Please note: If you cannot answer a question, you can earn 1 point by telling us a good *and* funny joke. You can only use 1 joke per page for no more than 3 total pages. So the maximum that you can earn is 3 points for 3 jokes over 3 pages.

## 1. Fundamentals

- A. What is different about the function of segment registers in protected mode versus real mode? (3 points)

**Real mode:** contain physical memory segment address

**Protected mode:** contain address of descriptor

- B. True or false: Protected mode segments have a fixed size of 4 GB. (3 points)

**False**

- C. Describe briefly the difference between polled and interrupt driven I/O. Give a technical example (i.e., not door bells or telephone ringers) of when you should use each one. (3 points)

**Polled I/O:** computer periodically checks device for input. Ex: Temperature sensing or other data acquisition application

**Interrupt driven I/O:** computer is alerted by a device when an input occurs. The computer can process other tasks until an input is received. Ex: network adapter generates interrupt when packet is received.

- D. What special data is stored in the first 1024 bytes of the PC's memory? (3 points)

**The Interrupt Vector Table**

- E. The SSE instructions introduced in the Pentium III processor are useful in speeding up what type of operations? (3 points)

**Floating point operations**

## 2. Interrupts

- A. List the three types of interrupts and give examples of each. (5 points)

**Hardware Interrupts:** Keyboard, Network adapter, etc

**Exceptions:** Divide by zero

**Software Interrupts:** INT 21h, the dos function dispatcher

- B. When you installed your own interrupt handlers, what exactly were you doing? (No code needed) (5 points)

**Placing the segment and offset address of my ISR in the appropriate row of the interrupt vector table.**

- C. What is the difference between ret and iret? (4 points)

**Iret pops the flag register from the stack.**

- D. What does “chaining” into an interrupt routine mean? (4 points)

**Calling the original ISR from your new ISR.**

- E. How many times will the keyboard IRQ be activated when you type the following string:

All your base are belong to us

Look carefully at the string before you answer. (6 points)

**30 characters, one key-down, one key-up, plus two for the shift  
 $2 \times 30 + 2 = 62$**

- F. True or false: INT8 is the same as IRQ8. (3 points)

**False**

G. Describe the function of the 8259 Programmable Interrupt Controller. (5 points)

It accepts interrupt signals from devices connected to the computer, places the corresponding interrupt number on the data bus, and triggers the interrupt input of the processor.

H. Consider the list of interrupts provided in Table 1 below. This table lists three interrupts, the time when the computer receives each interrupt's request for service, and how many time periods each corresponding interrupt service routine requires to completely execute.

Place one X in each column of Table 2 to indicate which task the computer will be executing for time periods 1 through 16.

*Hint: there should be one and only one X in each column. This is a simplified example of how interrupt scheduling really works. The emphasis here is on interrupt priority. You do not have to account for the time it takes to push and pop registers and flags. These times are already included in column three of Table 1. (8 points)*

Interrupt	Requested Start Time	Number of periods needed to run corresponding ISR
INT 47	Period 4	2 periods
INT 84	Period 2	4 periods
INT 103	Period 5	3 periods

Table 1

Time Period	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Main Program	X										X	X	X	X	X	X
INT 47				X	X											
INT 84		X	X			X	X									
INT 103								X	X	X						

Table 2

### 3. Video Graphics

- A. Many theaters are beginning to use digital projectors in their auditoriums rather than the traditional 35mm film projectors. These new digital projectors work on the same principle as the monitors in the ECE291 lab. But they must have much higher resolution since the display screen is not 19 inches, but 50 *feet* or more. To maintain the same quality image on such a large scale, these projectors typically have a resolution of 20,000 x 10,000 pixels. If we want to maintain a 24-bit color depth, how much video memory must a digital projector have available? (4 points)

$$20,000 \times 10,000 \times 3 = \mathbf{600,000,000 \text{ or } 600 \text{ MB}}$$

- B. In Mode 13h graphics (320x200 pixels with 256 colors), what is the correct offset from the beginning of the video memory for the pixel at row 100 and column 50? (4 points)

$$100 \times 320 + 50 = \mathbf{32,050}$$

- C. What is the main advantage of using Bresenham's Line Drawing Algorithm? (4 points)

Avoids the use of floating-point numbers, multiplication, division, and comparison with any number other than zero.

- or -

Can be implemented with shifts, integer adds, and comparisons with zero.

D. Using Bresenham's algorithm, figure out which pixels should be colored to represent a line from pixel (0,0) to pixel (6,4). Show all work, fill in the requested values and blanks in Table 3, and color the pixels representing the line in Table 4 (18 points)

$Dx = 6$                       Horizontal Additive Error =  $8$

$Dy = 4$                       Diagonal Additive Error =  $-4$

Starting Error (E) =  $2$

X	Y	E
0	0	$2$
1	$1$	$2 + (-4) = -2$
2	$1$	$-2 + 8 = 6$
3	$2$	$6 + (-4) = 2$
4	$3$	$2 + (-4) = -2$
5	$3$	$-2 + 8 = 6$
6	4	

Table 3

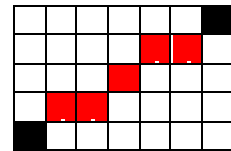


Table 4

## 4. MultiMedia eXtensions

- A. MMX Instructions primarily improve the performance of what category of operations? (2 points)

**Integer operations**

- B. Explain the difference between wraparound, saturation and signed saturation as they apply to packed bytes. (4 points)

**Wraparound:** results of operations that overflow lose the most significant (carry) bit

**Saturation:** results of operations that overflow are clamped at the largest representable value. In byte-sized operations, this value is FFh

**Signed saturation:** results of operations that overflow are clamped at the largest positive or smallest negative, depending on the direction of the overflow. In byte-sized operations, these values are 7Fh and 80h respectively.

- C. MMX instructions make use of the floating-point registers from the 80x87 floating-point unit. This technique is known as aliasing. What, if anything, do you have to do after executing an MMX instruction before executing a floating-point unit instruction? (4 points)

**EMMS**

- D. Write function `_HighlightRedBlue`, which is defined below. This function enhances a pixel's red and blue channels by adding a constant value to each appropriate color channel. Your code should update two pixels at once. When updating the pixels, you must use proper saturation when adding the highlight values. The video screen is 640x480, 32-bit color. (20 points)

```
void _HighlightImage(long *DestImage)
```

Purpose: Highlights the red and blue channels of a pixel

Input: `DestImage` - pointer to beginning of destination image in memory

Output: writes to destination image

```
HighlightByteZero dd 00000033h
```

```
HighlightByteOne dd 00003300h
```

```
HighlightByteTwo dd 00330000h
```

```
HighlightByteThree dd 33000000h
```

```
proc _HighlightImage
```

```
.DestImage arg 4
```

```
push edi
```

```
mov edi, [ebp+.DestImage]
```

```
mov ecx, 320*480
```

```
movd mm1, [HighlightByteZero] ; Red byte  
movd mm2, [HighlightByteTwo] ; Blue byte  
pxor mm3, mm3 ; Clear mm3 to use as mask  
por mm3, mm1  
por mm3, mm2 ; load mask into lower dword  
psllq mm3, 32 ; shift into high dword  
por mm3, mm1  
por mm3, mm2 ; load mask into lower dword
```

```
.HighlightLoop
```

```
movq mm0, [edi]
```

```
paddusb mm0, mm3 ; update pixels
```

```
movq [edi], mm0
```

```
add edi, 8
```

```
loop .HighlightLoop
```

```
pop edi
```

```
ret
```

```
endproc
```

## 5. Floating-Point Numbers

- A. Put the following decimal numbers in the *tiny* floating-point format. The tiny format is specified as follows: (10 points)

1 bit sign  
 4 bit exponent with bias of 7  
 6 bit mantissa with implicit first 1

**6.25      0 1001 100100**

**-5.375    1 1001 010110**

- B. Write a real-mode assembly procedure, CtoF, to convert from Celsius to Fahrenheit the single-precision temperature value passed in DX:AX. The result should be placed in the global memory variable pointed to by Temp. Recall the formula is:

$$F = \frac{9}{5}C + 32$$

(25 points)

Temp resd 1

```
thirtytwo    dd    32.0
ninefifths   dd    1.8
```

CtoF:

```
mov    [Temp], ax
mov    [Temp+2], dx
finit
fld    dword [Temp]
fmul  dword [ninefifths]
fadd  dword [thirtytwo]
fstp  dword [Temp]
ret
```