UNC Charlotte- Midterm Exam –October 15, 2003

1) E	6) B	11) B	16) D	21) C
2) D	7) D	12) D	17) C	22) A
3) D	8) E	13) B	18) C	23) B
4) E	9) D	14) A	19) D	24) E
5) E	10) B	15) E	20) B	25) D

26) (5 points). **(2 points for definition and 3 points for how it's used)**

The dynamic link holds the old value of the FB register when a function is called. It is saved in the activation record of the callee function on stack so that when we return to the caller function, we can still access its local variables. To accomplish this, the value is restored from the stack upon return.

27) (20 points) **(2 points for each box + 2 points for correct u0brg value)**

Clock	Actual Bit Rate	Percent Error	u0brg value
f1	**1199.62**	**-0.032%**	**1041 or NA**
f8	**1201.92**	**+0.16%**	**129**
f32	**1183.71**	**-1.36%**	**32**

Selecting the setting for the lowest percent error in the table above, what would be the valid value for u0brg when writing the c-code? u0brg **= 0x81; //(i.e. 129. Note u0brg is only 8 bits and can't hold 1041)**

28) (25 points) There are many solutions. Here's one that was hinted at:

(5 points for comments, 20 points for correct implementation)

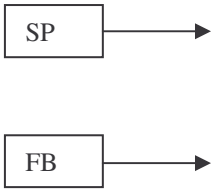
```
/*Author: S. Sair. 10/7/03. Solution to Q28 on ECE306 Fall'03 Midterm*/
int sum = 0; /*Holds the sum of numbers from 1 to 100.*/
int faulty_sum = 0; /* Holds the sum of the elements in x.*/
int missing, i;
for ( i=0; i < 100; i++ ){
   faulty_sum += x[i]; /* Note the corrupt value will be read as 0.*/
   sum += i+1; /*Note i varies from 0 to 99, hence the +1.*/
}
missing = sum - faulty_sum;
```

29) (20 points) (5 points for header, 15 points for correct code)

```
/* Author: S. Sair. 10/7/03.*/
/* Solution to Q29 on ECE306 Fall'03 Midterm. */
if(k){ /*or k!=0*/
  if(z){ /*or z!=0*/
      c++;
   }
   else{
      a++;
   }
}
else{
   if(x){ /*or x!=0*/
      b++;
   }
   else{
      d++;
   }
}
```

Note this isn't the only correct solution. Can build if statements using false predicates (i.e. if(!x) …).

30)



Address	Value	Description
0x07E1	XX	
0x07E2	XX	
0x07E3	XX	
0x07E4	XX	
0x07E5	XX	
0x07E6	0x01	n_l for factorial - 2nd call
0x07E7	0x00	n_h for factorial - 2nd call
0x07E8	XX	temp_l for factorial - 2nd call
0x07E9	XX	temp_h for factorial - 2nd call
0x07EA	0xF3	old FB_l for factorial - 2nd call
0x07EB	0x07	old FB_h for factorial - 2nd call
0x07EC	0x00	return address: factorial(L1_l)
0x07ED	0x04	return address: factorial(L1_m)
0x07EE	0x0F	return address: factorial(L1_h)
0x07EF	0x02	n_l for factorial - 1st call
0x07F0	0x00	n_h for factorial - 1st call
0x07F1	XX	temp_l for factorial - 1st call
0x07F2	XX	temp_h for factorial - 1st call
0x07F3	0xFC	old FB_l for factorial - 1st call
0x07F4	0x07	old FB_h for factorial - 1st call
0x07F5	0x00	return address: main(L2_l)
0x07F6	0x03	return address: main(L2_m)
0x07F7	0x0F	return address: main(L2_h)
0x07F8	XX	z_l
0x07F9	XX	z_h
0x07FA	0x02	m_l
0x07FB	0x00	m_h
0x07FC	0xFE	old FB_l for main
0x07FD	0x07	old FB_h for main
0x07FE	XX	