Problem 0 (1 point, 1 minute)

Put your login name on each page. Also make sure you have provided the information requested on the first page.

Problem 1 (6 points, 15 minutes)

Consider the following three machine instructions, which appear in memory starting at the address 0x00400000.

address (in hex)	*contents (in hex)*
00400000	12080002
00400004	3C11FFFF
00400008	08100004

Part a

"Disassemble" the instructions; that is, give an assembly language program segment that would be translated into the given machine language. You may use numeric rather than symbolic register names. A list of op codes (Figure A.19 from P&H) appears at the end of this exam.

Handle branches and jumps specially; where you would normally have a label, provide instead a hexadecimal byte address. For example, you should list a jump to the first instruction as

```
j 0x00400000
```

and represent a branch to the first instruction, say **bltz**, similarly as

```
bltz $9,0x00400000
```

Your login name: c s 6 1 c______

Part b

For each of the instructions indicate whether (a) it *must have contributed* an entry to the relocation table, (b) it *may have contributed* an entry to the relocation table, or (c) it *could not have contributed* an entry to the relocation table. Briefly explain your answers.

address (in hex)	*contents (in hex)*	*explanation of why this instruction must have, may have, or could not contribute relocation entry*
00400000	12080002	
00400004	3C11FFFF	
00400008	08100004	

Problem 2 (6 points, 20 minutes)

Consider the following C program segment.

```
int k, saved-k;
float x;
   ...
saved-k = k;
x = (float)k;
k = (int)x;
if (k == saved-k) {
   printf ("no change after conversion to float\n");
} else {
   printf ("change after conversion to float\n");
1
```

Recall that a cast converts the casted value to the given type. Thus if k contains the integer 3, the assignment

```
x = (float) k;
```

results in x containing the floating point value 3.0.

Assume for the following questions that an int and a float each use 4 bytes of memory, that a **double** uses 8 bytes of memory, and that a float and a **double** are stored using IEEE floating-point representation.

Part a

Find an int value k for which the above program segment produces the output

```
change after conversion to float
```

and give its hexadecimal representation.

Part b

Suppose that x in the above program segment was declared as **double.** Would the output still be the same, using your answer to part a? Briefly explain.

Your login name: cs61c-,

Part c

Give the *largest* (signed) hexadecimal integer value that k could contain and still produce the output

```
no change after conversion to float
```

Briefly explain your answer.

Part d

Give the 4-byte (single precision) IEEE floating-point representation (in hexadecimal) of your answer to part c. Show how you got your answer.

Problem 3 (7 points, 24 minutes)

Both parts of this question involve code from our solution to lab assignment 9, which appears at the end of this exam.

Part a

Suppose that the output buffer were redefined as follows:

```
buffer: .space 2
```

Below, describe clearly what other changes to the code are necessary to accommodate the smaller buffer (give line numbers **of** statements to be modified, and say what modifications are necessary).

Part b

Assume that the lab 9 code is correctly modified as specified in part a; recall that the modified buffer will hold at most one character at a time. Assume **also** that the following code is inserted at line 57.

```
        la $a0,str
        jal print
loop:   j loop
```

where **str** is defined as

```
str:   .asciiz "ABC"
```

Suppose now that a timer is started at the call to **print** that advances **1** time unit every ***assembly language*** instruction, and that approximately 1000 time units **pass** between when a character is stored into the transmitter data word and when the transmitter again becomes ready.

On the next page are three "traces" **of** execution behavior, labeled **A, B,** and **C.** Each line in each trace lists a label and the time that the corresponding labeled statement was executed. For 'example, the sequence

```
print       1
chkfull     7
intrp      16
```

means that the **lb** labeled by **print** was executed at time 1, the **lw** labeled by **chkfull** *(6* instructions further on) at time 7, and an interrupt occurred 9 instructions later. One of the traces represents the execution behavior that results from printing ***"ABC"*** as described above. Indicate which of the three traces most closely does so. *Also* describe, in terms relating to buffer management, interrupt handling, or transmitter operation, why each of the others ***does not*** represent the execution behavior that results from printing "ABC".

A	
B	
C	

Execution traces

Version A

print	1
chkfull	7
intrp	16
notEmpty	25
intDone	33.
print	43
chkfull	49, 51, 53, …, 1027, 1029
intrp	1030
notEmpty	1039
intDone	1047
print	1064
chkfull	1070, 1072, …, 2040, 2042
intrp	2044
notEmpty	2053
intDone	2061
print	2077
alldone	2080
loop	2081-inf

Version B

print	1
chkfull	7
intrp	16
notEmpty	25
intDone	1033
print	1043
chkfull	1049
intrp	1054
notEmpty	1063
intDone	2071
print	2081
chkfull	2087
intrp	2092
notEmpty	2101
intDone	3108
print	3118
alldone	3121
loop	3122-inf

Version C

print	1
chkfull	7
intrp	16
notEmpty	25
intDone	33
print	43
chkfull	49
print	59
chkfull	65, 67, …, 1027, 1029
intrp	1030
notEmpty	1039
intDone	1047
print	1065
alldone	1068
loop	1069-2043
intrp	2044
notEmpty	2053
intDone	2061
loop	2070-w