Name............................

Final - CS U380 - Dec. 18, 2008

No calculators, PDAs, etc.

1 Representation of numbers

a. **Assuming an 8-bit word length,** What is the two's-complement representation of -64(base 10)? Show all your work.

.

b. What is the 32 bit IEEE floating representation of $8\frac{1}{14}(base10)$?

.

c. What is the eight-bit ASCII representation of '$'?

For the following questions, your MIPS code should follow the callee-save conventions for register usage that we have been using in this course.

2 Pointers; Call and return

Translate to MIPS assembly language: (Note: you do NOT have to implement the functions g and h in part b.)
a.)

```
int
addVars(int *px, int *py, int *psum) {

    int val = *px + *py;
    *psum = val;
    return val;

}
```

.

b.)

```
int
fRec(int x, int y) {

    int temp = g(y);

    return temp + h(x + temp);

}
```

Hint: What register will you use for x? for *temp*?

.

Extra page, for your use

3 Pointers and arrays

Translate the following procedure, nextHigherChar, into MIPS assembly language, so that it can be called in any program:

```
void nextHigherChar(char *s) {

    char ch;

    while((ch = *s) != '\0') {
        ch = ch + 1
        *s = ch;

        /* increment s */
        s++;

    } /* end while */
}
```

.

Extra page, for your use

4 Recursion

Calculate the length of a string. Use a method similar to the one for calculating the length of a list in the Felleisen book.

```
/* Translate literally- don't eliminate the recursion. */
int strlen(char *s) {

    if(*s == '\0') {
        return 0;
    } else {
        s = s + 1;
        return 1 + strlen(s);
    }
}
```

.

Extra page, for your use

5 Arrays and function values

Translate to a function in MIPS assembly language.

```
int countEqualElements(int a[], int b[], int len) {

    int count = 0;
    int i;

    for(i = 0; i < len; i++) {
        if(a[i] == b[i])
            count++;
    }

    return count;
}
```

6 Representation of instructions

For this question, write any addresses that are part of your answer in hexadecimal.

a.

Write one line of assembly language that will assemble into the following instruction:

6 bits	5 bits	5 bits	5 bits	5 bits	6 bits
001000	10000	01000	11111	11111	111111

.

b.

Write one line of assembly language that will assemble into the following instruction:

6 bits	5 bits	5 bits	5 bits	5 bits	6 bits
000000	10000	10001	01000	00000	101010

.

c.

Write one line of assembly language that will assemble into the following instruction:

6 bits	5 bits	5 bits	5 bits	5 bits	6 bits
000010	00000	00000	00000	00000	001001

.