Name: _____

CS 33003 **Exam #3** **Comp Org & Assm Lang**

Friday 14 November 1997

1. (20 points) Given the following C code fragment:

```
i = 1;
while ( i <= 100)
        i = i * i;
```

Write an equivalent RISC LOAD/STORE assembly language program, with all necessary segments, code, and data structures (store final value of "i" in memory).

```
.bss
i:      .reserve 4
.text
        mov     R1, #1
test:   ble     R1,#100,while
        jump    end
while:  mpy     R1,R1,R1
        jump    test
end:    store   i,R1
```

2. (15 points) Write a small piece of RISC LOAD/STORE assembly language code that illustrates a forward reference and a backward reference, and indicate which is which. Then explain which is harder for the assembler to process, and why.

```
        jump    fwd     This jump makes a forward reference, which is a
        …               reference to a label that hasn't been defined yet.
fwd:    …
        …
bck:    …
        …               This jump makes a backward reference, which is a
        jump    bck     reference to a label that has already been defined.
```

Forward references are harder to process, because the assembler can't fill in the address in the instruction, since it doesn't know yet what address the label refers to.

Name: _____

3. (12 points) What are the potential problems (more than one) with the following assembly language code fragment, which you can assume is part of a larger program? (For convenience, I have shown the code in several columns, but you can assume that the code in column 3 follows the code in column 2, which follows the code in column 1.)

```
main:   …                   sub1:   …                   sub2:   …
        mov   R2,100                jump  sub2                  mov   R2,300
        jump  sub1          s2ret:  …                           add   R1,R1,R2
s1ret:  …                           jump  s1ret                 …
        add   R2,R2,R1                                          jump  s2ret
        …
```

This was intended to be a question on subroutines, with the following answers being the ones I had in mind:

(i) Subroutines sub1 can only be called from that one place in main, and sub2 can only be called from that one place in sub1; if they are called from anywhere else, they won't be able to return to the instruction immediately following the call.

(ii) Subroutines sub1 and sub2 aren't saving and restoring the contents of any registers they use, so sub2 may be corrupting the values in R1 and R2 which main used before the call to sub1, and assumed would be the same after the call to sub1.

However, it was also pointed out that: (i) there's an infinite loop, since once sub1 returns to s1ret, it soon goes back into sub1, then back to s1ret, etc., and (ii) the mov instructions should be loading a value with "#", else they'll load the data at that address, rather than loading that immediate value.

4. (18 points) Show the **5-bit** representation of each of the decimal values below in each of the specified formats. (18 points)

Value	Signed Magnitude	Excess 16	2's Complement
15	**01111**	**11111**	**01111**
-4	**10100**	**01100**	**11100**
-15	**11111**	**00001**	**10001**

Name: _____

5. (20 points) Convert 22.625 to IEEE floating point format, showing your work as much as necessary so that if your answer is wrong I can give you partial credit.

$22_{10} = 16 + 4 + 2 = 10110_2$ **$0.625_{10} = 1/2 + 1/8 = 0.101_2$**

$22.625_{10} = 10110.101_2 = 1.0110101_2 \times 2^4 = 1.1110011_2 \times 2^{131-127}$

0 100000011 01101010000000000000000

6. (15 points) In the SPARC CPU,

   a. What does the instruction "ld [%l2+%l3],%l4" do?

   Adds the contents of local register 2 to the contents of local register 3, treats the result as an address, and loads the contents of the memory location specified by that address into local register 4.

   b. Give an example of a synthetic instruction (just an instruction, not its implementation).

   clr, inc, mov, set

   c. In what way is the SPARC a "RISC" CPU?

   It has a reduced instruction set (not even multiplication and division), very few addressing modes, all instructions execute in one cycle, etc.

Name: _____