Name:

Question	Possible	Score
1	15	12
2	20	19
3	15	15
4	15	11
5	15	13
6	20	16
Total	100	86

General information that may be useful sometime during test:

Table of Powers of Two

N	2^N		N	2^N		N	2^N		N	2^N
0	1		8	256		16	65,536		24	16,777,216
1	2		9	512		17	131,072		25	33,554,432
2	4		10	1,024		18	262,144		26	67,108,864
3	8		11	2,048		19	524,288		27	134,217,728
4	16		12	4,096		20	1,048,576		28	268,435,456
5	32		13	8,192		21	2,097,152		29	536,870,912
6	64		14	16,384		22	4,194,304		30	1,073,741,824
7	128		15	32,768		23	8,388,608		31	2,147,483,648

1. The general information question:
   a) In the MIPS architecture, what happens on a branch-to-subroutine? This is a subroutine linkage question, not a parameter passing question. That is, identify what happens to the PC and also to the return address.

jal target → $rA <= PC+4
PC <= target address

PC will get the address of the subroutine and the return address is stored in register $rA.

   b) describe the activities associated with (that is, give the RTL for…) the `add` instruction `(add $s1,$s2,$s3)`.

The values stored in registers $s2 and $s3 are added, and the sum is stored in the destination register $s1.

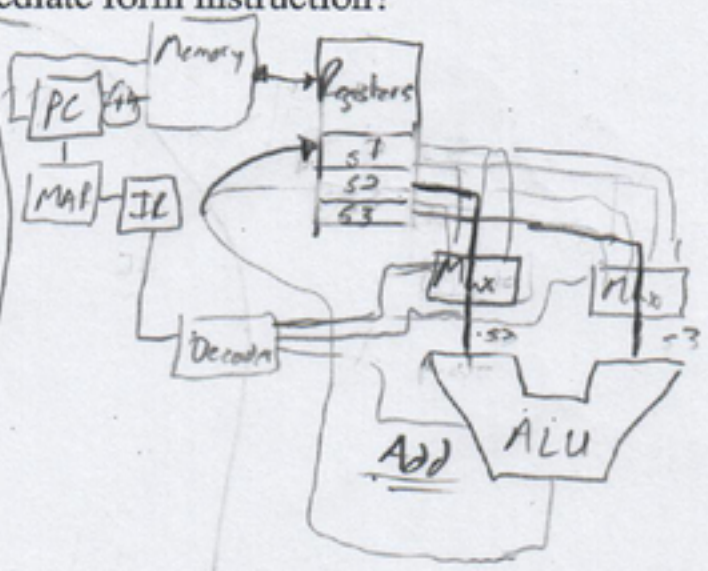
Fetch
PC <= instruction
MAR <= PC
IR <= m[MAR]
PC <= PC+4

Decode
No register transfer

Execute:
s1 <= m[s2] + m[s3]
contents of s2
contents of s3

   c) Where is the immediate information stored for an immediate form instruction?

In memory, SIMM or UIMM
in register



2. The floating point question: Use the IEEE 32 bit floating point system for the answers to this question. The first part of the question is value representation. The second part is doing the math with IEEE system. Remember that the system has the following characteristics: Radix of system: 2; number of exponent bits: 8; exponent representation method: excess-127; number of mantissa bits: 23 (24 with hidden bit); mantissa range for normalized numbers: $1 \leq$ mantissa < 2; mantissa stored using hidden bit technique.

$63 = 64 + 16 + 2 + 1$

a) Give the bit pattern (in the IEEE system) for the value $83\ ^{21}/_{64}$.

Exponent = 6+127 = 133
Sign = 0
1.010011×2^6

Sign | 133 | Mantissa
0 | 10000101 | 01001100000000000000

b) What is the base 10 representation of the value represented by the bit pattern
1 10000101 11111111000000000000000 ?

128+5=133
-127
2^6
1.1111111×2^6
$\frac{1}{2}+\frac{1}{4}=\frac{3}{4}$
$\Rightarrow -127.75$

c) multiply the values represented by the two patterns (give results in same format):
1) 1 10000110 10101000000000000000000
2) 1 10000011 01001000000000000000000

0 10001011 00011111010000000000

128+6=134
-127
7
$\Rightarrow 1.1010100 = -(128+64+20) = -212$

128+3=131
-127
4
$\Rightarrow 1.01001 = -(20.5) = -20.5$

212
x 20.5
1060
000
424
4346.0

4346 = 4096 + 128 + 64 + 32 + 16 + 8 + 2

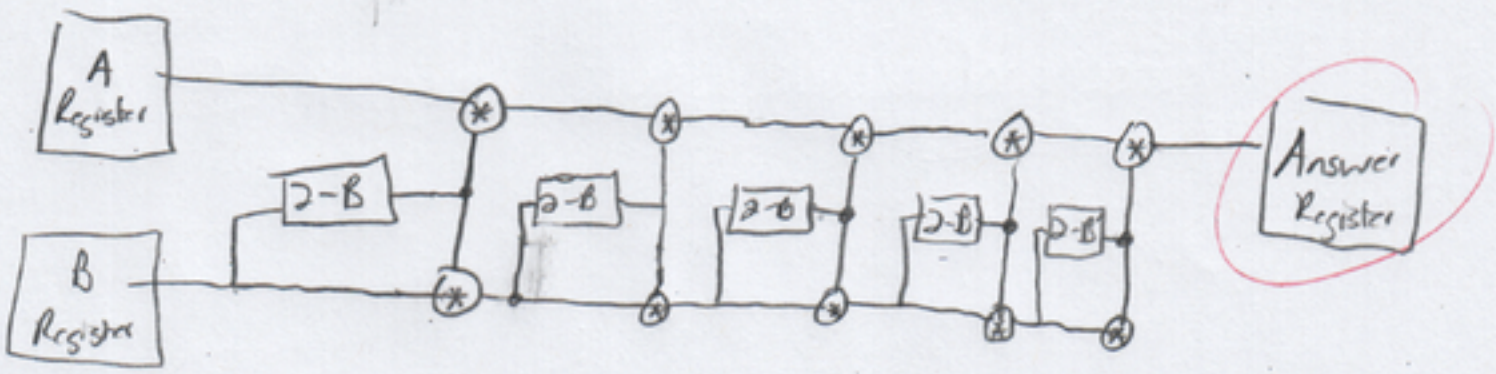
4096
+128
4224
122
-64
58
-32
26
-16
10

10000111111010

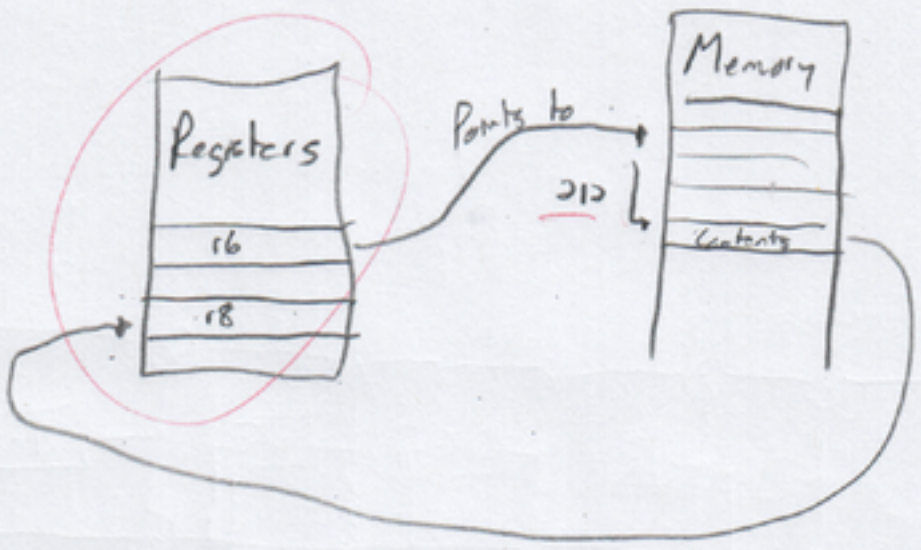
b + 127 = 134

10001011

3. Divide question: Assume that you have a bunch of IEEE Floating Point Multipliers. In the space below, show how to connect the multipliers and floating point subtract units (to develop 2 – B) in order to create a divide function. Assume 5 stages is enuf. Do it in the classical block diagram method – not state machine driven, but just functionally driven.



4: RTL question: Give an RTL description of the transfers needed to carry out the work of an `lw r8,212(r6)` instruction in the MIPS architecture. In other words, what are the necessary register transfers required to do the instruction. Note that we are not trying to be tricky here and specify pipelining or anything like that. This is just a what-does-the-instruction-do kind of answer.



NO?!

The base address in memory is stored in register r6. We are accessing the memory "array element" that is 212 bytes offset from that base address and loading an entire word worth of information from that location into register r8.

Fetch

PC <= instruction

MAR <= PC

IR <= m[MAR]

PC <= PC+4

Decode

Decodes the instruction but no transfer in register

Execute

r8 <= m[m[r6] + 212]

one too many here!

not quite correct

contents of IR

contents of this memory location

5: VHDL Question 1: Below is the code for a standard 'D' flip-flop. But there are some errors in the code. Identify & correct the errors. (Note: the errors are VHDL errors, not logic errors.)

```
library IEEE;
use IEEE.STDLOGIC_1164.all;

entity DFF is
  port (
    D   : in  STD_LOGIC;
    SET : in  STD_LOGIC;
    CLR : in  STD_LOGIC;
    CLK : in  STD_LOGIC;
    Q   : out STD_LOGIC;
    QN  : out STD_LOGIC;
  );
end entity DFF

architecture FOR_TEST of DFF is

signal INTERNAL_D : BIT;

begin

THE_PROC:
  process ( CLK, D ) is
  begin
    if SET == '1' then
      INTERNAL_D <= '1';
    else if CLR = '1' then
      INTERNAL_D <= '0';
    else if RISING_EDGE ( CLK ) then
      INTERNAL_D <= D;
    end if;

    wait for 5 ns;

    Q  <= INTERNAL_D;
    QN <= not INTERNAL_D;

  end;

end architecture FOR_CLASS;
```

std_logic;

SET, CLR

==

elsif

elsif

(CLK)

Don't think you can do this assignment

end process;

wait for 5 ns;

wait for 5 ns;

TEST

6: VHDL Question 2: Elsewhere in the system code has been established to generate the product of two 24-bit mantissas. (In other words, somebody else did the multiply of two 24-bit mantissas, and you don't have to do that.) This answer is available in the vector ANS(47 downto 0). The creator of this code wishes to generate his final value (the result of the multiplication of two 32-bit floating point numbers) as SIGN & EXP & MANT. SIGN is a single bit STD_LOGIC value. EXP is to be an 8 bit STD_LOGIC_VECTOR. At the start of this process, the exponent is represented as BEGIN_EXP, which is an 8 bit STD_LOGIC_VECTOR. Note that something should be done with the value represented as BEGIN_EXP to generate the correct value for EXP in the post-normalization process. Finally, MANT is a 23 bit STD_LOGIC_VECTOR. In the space provided below, give the VHDL description of the logic to correctly obtain the MANT bits from the ANS bits and the EXP value from BEGIN_EXP. (Note that MANT of this question is 23 bits, not 24 bits.)

1 0.
47 ' 45

Truncating

```
MANT <= ANS(46 downto 24) when ANS(47) = '1' else ANS(45 downto 23);
EXP2 <= BEGIN_EXP + 1;
EXP  <= EXP2 when ANS(47) = '1' else BEGIN_EXP;
EVAL <= S & EXP & MANT
```

close. but how to round?