Decoding MIPS Instructions

Every MIPs instruction is represented with 32 bits! They come in three formats:

R-Instruction format (register-to-register) *Examples: add, and, sll, slt, jr*

op code	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

I-Instruction Format (register immediate) Examples: addiu, andi, beq, bne

op code	rs	rt	immediate
6 bits	5 bits	5 bits	16 bits

J-Instruction Format (jump format) For j and jal

op code	address
6 bits	26 bits

Here's what each field in the formats mean:

opcode	indicates operation, or arithmetic family of operations (for opcode 0, which is R-type)
funct	indicates specific operation within arithmetic family of operations
rs, rt, rd	for R-type, rs and rt are sources with rd as destination – rules vary for other formats!
shamt	shift amount for instructions that perform shifts
immediate	Relative address or constant, will be 0 or sign-extended to 32 bits
address	Absolute address

See the MIPS Green Sheet for more details!

Question 1: Convert `addi $t1, $t0, 5` to its HEX representation.

Format: addi $rt, $rs, imm ---> opcode(addi) = 001000, $t0 = 01000, $t1 = 01001
0x21090005 or 0b001000 | 01000 | 01001 | 0000000000000101

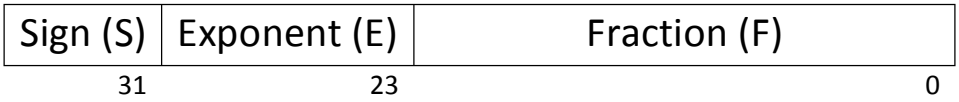
Question 2: Decode the following program and describe its function.
This function returns the larger number of $a0 and $a1.

Address	Instruction	Decoded Instruction
0x00	0x0085402A (0b00000000100001010100000000101010)	slt $t0, $a0, $a1
0x04	0x11000002 (0b00010001000000000000000000000010)	beq $t0, $0, 2
0x08	0x00A01020 (0b00000000101000000001000000100000)	add $v0, $a1, $0
0x0c	0x03E00008 (0b00000011111000000000000000001000)	jr $ra
0x10	0x00801020 (0b00000000100000000001000000100000)	add $v0, $a0, $0
0x14	0x03E00008 (0b00000011111000000000000000001000)	jr $ra

Floating Point Numbers (IEEE Standard 754)

$$\text{FP value} = (-1)^{S} \times (1 + F) \times 2^{(E - \text{bias})}$$

Single Precision FP (32 bit)



Single Precision: S = 1 bit, E = 8 bits, F= 23 bits, bias = 127.
Double Precision: S = 1 bit, E = 11 bits, F= 52 bits, bias = 1023.

Special Values:

±Zero: E = 0, F = 0

NaN: E = 255, F ≠ 0

±Infinity: E=255, F = 0

Denormalized: E = 0, F ≠ 0

Question 3: Why do we use a bias?

We use bias because we wish to represent both tiny and large real numbers. Subtracting bias from our exponent value will share the range we can represent to both tiny (negative E) and large (positive E) real numbers.

Question 4: Convert the single precision FP representation 0xC0B40000 to decimal.

0xC0B40000 = 0b1 | 10000001 | 01101000000000000000000

0b10000001 = 0d129 ---> E = 129 – 127 = 2

-1.01101×2^2

$= -(2^0 + 2^{-2} + 2^{-3} + 2^{-5}) \times 2^2$

$= -(2^2 + 2^0 + 2^{-1} + 2^{-3})$

$= -(4 + 1 + 0.5 + 0.125)$

$= -5.625$

Now we know how to convert from FP representations to decimals, how about the other way around? Google is always your best friend. For example, try this website: http://www.cs.cornell.edu/~tomf/notes/cps104/floating.html#dec2hex