(a) (1 point) The IEEE double-precision floating point format consists of a 1-bit sign S, an 11-bit biased exponent E with a bias of 1023 and a 52-bit fraction F, as shown below:

S	E	F

Consider the following two IEEE double-precision floating point operands *x* and *y* (specified in hex): *x* = `C014800000000000` , *y* = `C024100000000000`. Show the computations, *performed in binary*, to calculate the sign, biased exponent and fraction of the quantity *2x – 4y*. Also, give the IEEE double-precision floating point representation (specified in hex) for this quantity.

(b) (0.5 points) Explain the concept of *Round to Nearest Even (RNE)* and explain why it is often used in floating point computations.

(c) (1 point) Consider the following sequence of decimal word address references on a certain computer:

30, 86, 53, 61, 29, 37, 30, 45, 6, 22, 14, 6, 53, 29, 78, 22, 70, 61, 54, 62, 45, 30, 21, 77, 85, 46, 33

Indicate if each address reference is a hit or a miss, and also show the final cache contents for an 8-way set-associative cache with one-word blocks and a total size of 64 words. Assume that LRU (least recently used) replacement is used, and that the cache is initially empty.

(d) (1.5 points) A 32-bit processor contains 32 32-bit registers, r0 through r31, and condition code bits C (carry), V (signed overflow), Z (zero) and N (negative). Some instruction encodings are given on the next page. Of these, only the SUB instruction affects the values of the condition codes. Source and destination registers are specified in the instruction encodings using a 5-bit unsigned binary representation of the register number. Any operand specified as a 16-bit or 22-bit immediate value is sign-extended to 32 bits prior to performing the operation. Note that a SUB instruction calculates either (contents of rs1) – (contents of rs2) or (contents of rs1) – (sign-extended immediate operand). Also, on a SUB instruction, C = 1 if there is a borrow and C = 0 otherwise.

Consider the following machine language program specified in hex notation: (Each line contains one 32-bit instruction; spaces are inserted within each line in order to improve readability.)

```
5B 00 00 37
13 59 00 3F
13 DA 60 00
F8 00 00 00
```

Give the corresponding assembly language program. (Use decimal notation for any immediate values.) Also, give the 32-bit contents (specified in hex) of registers r12 through r15, inclusive, and the values of the condition code bits C, V, Z and N after each instruction has been executed by the processor. (Use the symbol x to denote a bit or hex digit which is unknown or uninitialized.)

SUB* – Subtraction

*This instruction sets the condition code bits.

Assembly code notation

a) SUB rdst, rs1, rs2
b) SUB rdst, rs1, #immed16

Instruction encoding

a)

31 ··· 27	26 ··· 22	21 ··· 17	16	15 ··· 11	10 ··· 0
00010	rdst	rs1	0	rs2	000 0000 0000

b)

31 ··· 27	26 ··· 22	21 ··· 17	16	15 ··· 0
00010	rdst	rs1	1	immed16

HLT – Halt

Assembly code notation

HLT

Instruction encoding

31 ··· 27	26 ··· 0
11111	000 0000 0000 0000 0000 0000 0000

LDI – Load immediate

Assembly code notation

LDI rdst, #value

Instruction encoding

31 ··· 27	26 ··· 22	21 ··· 0
01011	rdst	immed22

NOT – Bit-wise logical complement

Assembly code notation

a) NOT rdst, rs1

Instruction encoding

31 ··· 27	26 ··· 22	21 ··· 17	16 ··· 0
00101	rdst	rs1	0 0000 0000 0000 0000

XOR – Bit-wise logical exclusive-OR

Assembly code notation

a) XOR rdst, rs1, rs2
b) XOR rdst, rs1, #immed16

Instruction encoding

a)

31 ··· 27	26 ··· 22	21 ··· 17	16	15 ··· 11	10 ··· 0
00110	rdst	rs1	0	rs2	000 0000 0000

b)

31 ··· 27	26 ··· 22	21 ··· 17	16	15 ··· 0
00110	rdst	rs1	1	immed16