NUMBER SYSTEMS

and

DATA REPRESENTATION

for

COMPUTERS (PROBLEM ANSWERS)

05 March 2008

Table of Contents

> There are three kinds of mathematicians: those who can count, and those who can't.[1]

[1] Attributed to the famous mathematician E. Bombieri. Albert R. Meyer, "Problems on Equational Proof Systems", M.I.T. (18 October 1998). http://courses.csail.mit.edu/6.044/fall98/sep25.html Visited 12 January 2008.

Conversion Between Binary and Hexadecimal Answers

Convert Single Digit Hexadecimal to Binary by Inspection

Problem 1. 0010_2

Problem 2. 0111_2

Problem 3. 1000_2

Problem 4. 1011_2

Problem 5. 1101_2

Problem 6. 1111_2

Convert Binary to Hexadecimal

Problem 7. D_{16}

Problem 8. 9_{16}

Problem 9. 7_{16}

Problem 10. 5_{16}

Problem 11. 101_{16}

Problem 12. $6B815_{16}$

Convert Hexadecimal to Binary

Problem 13. $0101:1110_2$

Problem 14. $1101:1010:1011_2$

Problem 15. $1000:0011:1100:0010:1111:1010_2$

Problem 16. $0001:0000:1110:1101:0111_2$

Data Representation Answers

Character Code Problems

Problem A1. 2 Zoos

Character	**2**	♭	**Z**	**o**	**o**	**s**
EBCDIC	F2	40	E9	96	96	A2
Binary	1111:0010	0100:0000	1110:1001	1001:0110	1001:0110	1010:0010

Character	**2**	♭	**Z**	**o**	**o**	**s**
ASCII	32	20	5A	6F	6F	73
Binary	0011:0010	0010:0000	0101:1010	0110:1111	0110:1111	0111:0011

Problem A2. 1 Bill

Character	**1**	♭	**B**	**i**	**l**	**l**
EBCDIC	F1	40	C2	89	93	93
Binary	1111:0001	0100:0000	1100:0010	1000:1001	1001:0011	1001:0011

Character	**1**	♭	**B**	**i**	**l**	**l**
ASCII	31	20	42	69	6C	6C
Binary	0011:0001	0010:0000	0100:0010	0110:1001	0110:1100	0110:1100

Problem A3. Hi ♭Ho ♭10 ♭lO?

Notes:

- “Hi” are letters.
- “Ho” are letters.
- “10” are numbers.
- “lO” are letters, followed by punctuation.

Character	**H**	**i**	♭	**H**	**o**	♭
EBCDIC	C8	89	40	C8	96	40
Binary	1100:1000	1000:1001	0100:0000	1100:1000	1001:0110	0100:0000
ASCII	48	69	20	48	6F	20
Binary	0100:1000	0110:1001	0010:0000	0100:1000	0110:1111	0010:0000

Character	**1**	**0**	♭	**l**	**O**	**?**
EBCDIC	F1	F0	40	93	D6	6F
Binary	1111:0001	1111:0000	0100:0000	1001:0011	1101:0110	0110:1111
ASCII	31	30	20	6C	4F	3F
Binary	0011:0001	0011:0000	0010:0000	0110:1100	0100:1111	0011:1111

Problem A4. 2ZS50Oo ␣1li.

- 2ZS5 Number, Letter, Letter, Number
- 0Oo Number, Letter, Letter
- **1li.** Number, Letter, Letter, Punctuation

Character	**2**	**Z**	**S**	**5**	**0**	**O**
EBCDIC	F2	E9	E2	F5	F0	D6
Binary	1111:0010	1110:1001	1110:0010	1111:0101	1111:0000	1101:0110
ASCII	32	5A	53	35	30	4F
Binary	0011:0010	0101:1010	0101:0011	0011:0101	0011:0000	0100:1111

Character	**o**	␣	**1**	**l**	**i**	**.**
EBCDIC	96	40	F1	93	89	75
Binary	1001:0110	0100:0000	1111:0001	1001:0011	1000:1001	0111:0101
ASCII	6F	20	31	6C	69	2E
Binary	0110:1111	0010:0000	0011:0001	0110:1100	0110:1001	0010:1110

Problem A5. Code the following in hexadecimal using ASCII and then convert to binary.

Character	O	v	e	n
ASCII Hex	**4F**	**76**	**65**	**6E**
ASCII Binary	0100:1111	0111:0110	0110:0101	0110:1110
EBCDIC Hex	D6	A5	85	95
EBCDIC Binary	1101:0110	1010:0101	1000:0101	1001:0101

Character	␣	H	o	t
ASCII Hex	**20**	**68**	**6F**	**74**
ASCII Binary	0010:0000	0110:1000	0110:1111	0111:0100
EBCDIC Hex	40	88	96	A3
EBCDIC Binary	0100:0000	1000:1000	1001:0110	1010:0011

Problem A6. Code the following in hexadecimal using EBCDIC and then convert to binary.

Character	1	0	␣	i
ASCII Hex	31	30	20	69
ASCII Binary	0011:0001	0011:0000	0010:0000	0110:1001
EBCDIC Hex	**F1**	**F0**	**40**	**89**
EBCDIC Binary	1111:0001	1111:0000	0100:0000	1000:1001

Character	s	␣	l	o
ASCII Hex	73	20	6C	6F
ASCII Binary	0111:0011	0010:0000	0110:1100	0110:1111
EBCDIC Hex	**A2**	**40**	**93**	**96**
EBCDIC Binary	1010:0010	0100:0000	1001:0011	1001:0110

Problem A7. Convert the following to hexadecimal and then to characters using EBCDIC code.

Binary	1111:0001	0100:0000	1101:0001	1010:0100	1001:0100	1001:0111
EBCDIC Hex	F1	40	D1	A4	94	97
Character	1	␣	J	u	m	p

Jubilation Code and Gran Zeff Code Problems

Problem B1. Code the word a del f o**s** (brother) from SECRET SYMBOL to JUBILATION CODE hexadecimal, and then to JUBILATION CODE binary.

a	d	e	l	f	o	**s**
A1	A4	B1	C3	F2	D3	E2
1010:0001	1010:0100	1011:0001	1100:0011	1111:0010	1101:0011	1110:0010

Problem B2. Code the word d w r o n (gift) from SECRET SYMBOL to GRAN ZEFF CODE hexadecimal, and then to GRAN ZEFF CODE binary.

d	w	r	o	n
48	99	85	77	75
0100:1000	1001:1001	1000:0101	0111:0111	0111:0101

Problem B3. Code the word s o f os (wise) from SECRET SYMBOL to UNICODE hexadecimal, and then to UNICODE binary.

s	o	f	o	s
03C3	03BF	03C6	03BF	03C2
0000:0011: 1100:0011	0000:0011: 1011:1111	0000:0011: 1100:0110	0000:0011: 1011:1111	0000:0011: 1100:0010

Problem B4. Code the word k a r d i a (heart) from SECRET SYMBOL to UNICODE hexadecimal, and then to UNICODE binary.

k	a	r	d	i	a
03BA	03B1	03C1	03B4	03B9	03B1
0000:0011: 1011:1010	0000:0011: 1011:0001	0000:0011: 1100:0001	0000:0011: 1011:0100	0000:0011: 1011:1001	0000:0011: 1011:0001

Problem B5. Code the JUBILATION CODE binary into JUBILATION CODE hexadecimal, and then to SECRET SYMBOL. (death)

1011:0100	**1010:0001**	**1101:0001**	**1010:0001**	**1110:0100**	**1101:0011**	**1110:0010**
B4	A1	D1	A1	E4	D3	E2
q	a	n	a	t	o	s

Problem B6. Code the GRAN ZEFF CODE binary into GRAN ZEFF CODE hexadecimal, and then to SECRET SYMBOL. (wicked)

0111:1000	**0111:0111**	**0111:0101**	**0101:0111**	**1000:0101**	**0111:0111**	**1000:0110**
78	77	75	57	85	77	86
p	o	n	h	r	o	s

Problem B7. Code the UNICODE binary into UNICODE hexadecimal, and then into **SECRET** SYMBOL. (nation)

0000:0011: 1011:0101	0000:0011: 1011:1000	0000:0011: 1011:1101	0000:0011: 1011:1111	0000:0011: 1100:0010
03B5	03B8	03BD	03BF	03C2
e	q	n	o	s

Conversion to and from Decimal Answers

Conversion to Decimal Answers

Problem A1. 6_{10}

Problem A2. 22_{10}

Problem A3. 429_{10}

Problem A4. 121_{10}

Problem A5. 180_{10}

Problem A6. $11{,}359_{10}$

Problem A7. 197_{10}

Problem A8. 1551_{10}

Common Answer Table for

Conversion from Decimal: Problems B1 through B6

Between Base 4, Binary, and Hexadecimal: Problem B7

Between Octal and Binary: Problem B8

	Column →	A	B	C	D
Row	Decimal	Binary	Hexadecimal	Base 4	Octal
1	31	1:1111	1F	133	37
2	99	110:0011	63	1203	143
3	165	1010:0101	A5	2211	245
4	169	1010:1001	A9	2221	251
5	257	1:0000:0001	101	10001	401
6	445	1:1011:1101	1BD	12331	675
7	479	1:1101:1111	1DF	13133	737
8	708	10:1100:0100	2C4	23010	1304

	Column →	A	B	C	D
Row	Decimal	Binary	Hexadecimal	Base 4	Octal
1	31	B1	B3	B5	B6
2	99	B1	B3	B5	B6
3	165	B1	B3	B5	B6
4	169	B1	B3	B5	B6
5	257	B4	B2	B7	B8
6	445	B4	B2	B7	B8
7	479	B4	B2	B7	B8
8	708	B4	B2	B7	B8

Number Systems Conversion Answers

Problem 1. Convert the following decimal numbers to binary.

Problem	Decimal	Hexadecimal	Binary
1a	535	217H	10 0001 0111
1b	240	F0H	1111 0000
1c	61	3DH	11 1101
1d	202	CAH	1100 1010

Problem 2. Convert the following decimal numbers to hexadecimal.

Problem	Decimal	Hexadecimal	Binary
2a	37	25H	10 0101
2b	222	DEH	1101 1110
2c	168	A8H	1010 1000
2d	959	3BFH	11 1011 1111

Problem 3. Convert the following numbers to decimal.

Problem	Binary	Hexadecimal	Decimal
3a	1101 0111	D7H	215
3b	0111 1001	79H	121
3c	1110 1011	EBH	235
3d	111 1111	7FH	127

String Function Answers

Problem 1. Length. Let

x = | B | l | e | s | s | e | d | ␣ | a | r | e | ␣ | t | h | e | ␣ | p | o | o | r | l |

Find k = LEN(x) = 20

Problem 2. Concatenation. Let

x = | B | e | ␣ | l |

y = | P | r | e | p | a | r | e | d | l |

Concatenate x and y. z = x & y.

z = | B | e | ␣ | P | r | e | p | a | r | e | d | l | | | | | | | | | |

Problem 3. Left. Let

x = | B | l | e | s | s | e | d | ␣ | a | r | e | ␣ | t | h | e | ␣ | p | o | o | r | l |

Find y = Left(x,5).

y = | B | l | e | s | s | l | | | | | | | | | | | | | | | |

Problem 4. Right. Let

x = | B | l | e | s | s | e | d | ␣ | a | r | e | ␣ | t | h | e | ␣ | p | o | o | r | l |

Find y = Right(x,8).

y = | t | h | e | ␣ | p | o | o | r | l | | | | | | | | | | | | |

Problem 5. MID. Let

x = | B | l | e | s | s | e | d | ␣ | a | r | e | ␣ | t | h | e | ␣ | p | o | o | r | l |

Find y = MID(x,14,2)

y = | h | e | l | | | | | | | | | | | | | | | | | | |

Problem 6. Comparison. Let

x = | B | l | e | s | s | e | d | ␣ | a | r | e | ␣ | t | h | e | ␣ | p | o | o | r | l |

y = | B | l | e | s | s | e | d | ␣ | a | r | e | ␣ | t | h | e | ␣ | m | e | e | k | l |

If x < y

Answer is “False”.

Problem 7. Search.

x = | B | e | h | o | l | d | , | ␣ | I | ␣ | b | r | i | n | g | ␣ | g | l | a | d | l |

y = | h | o | l | d | l |

k = FIND(x,y) = 3

Problem 8. Search.

x =	t	i	d	i	n	g	s	.	␣	Y	o	u	␣	s	h	a	l	l	l
y =	y	o	u	l															

Search only for whole words; permit upper or lower case.

k = FIND(x,y) = 10

Problem 9. Compound Function

x =	B	l	e	s	s	e	d	␣	a	r	e	␣	t	h	e	␣	p	o	o	r	l
y =	P	r	e	p	a	r	e	d	l												

Find z = LEFT(x,5) & MID(x,13,3) & y

z =	B	l	e	s	s	␣	t	h	e	␣	P	r	e	p	a	r	e	d	l		

Problem 10. Compound Function

x =	B	e	␣	b	r	a	v	e	.	␣	B	e	␣	s	t	r	o	n	g	.	␣
	B	e	␣	r	e	a	d	y	.	␣	B	e	␣	a	l	e	r	t	.	l	
y =	B	e	l																		

Find

z = RIGHT(x, LEN(x)–LEN(y)–FIND(x,y, FIND(x,y, FIND(x,y)+LEN(y))+LEN(y)))

z =																				

Solution:

z = RIGHT(x, LEN(x)–LEN(y)–FIND(x,y, FIND(x,y, **FIND(x,y)+LEN(y)**)+LEN(y)))

j = FIND(x, y) = 1

k = **LEN(y) = 2**

z = RIGHT(x, LEN(x)–LEN(y)–FIND(x,y, FIND(x,y, **1**+**2**)+LEN(y)))

z = RIGHT(x, LEN(x)–LEN(y)–FIND(x,y, FIND(x,y, **3**)+LEN(y)))

FIND(x,y, 3) = 11

z = RIGHT(x, LEN(x)–LEN(y)–FIND(x,y, **11**+LEN(y)))

z = RIGHT(x, LEN(x)–LEN(y)–FIND(x,y, 11+**2**))

z = RIGHT(x, LEN(x)–LEN(y)–FIND(x,y, 13))

m = **LEN(x) = 40**

z = RIGHT(x, **40**–**2**–FIND(x,y, 13)) = RIGHT(x, 38–FIND(x,y, 13))

FIND(x,y, 13) = **22**

z = RIGHT(x, 38–22) **= RIGHT(x, 16)**

z =	r	e	a	d	y	.	␣	B	e	␣	a	l	e	r	t	.	l

Hexadecimal Addition Answers

Problem 1. 1110_{16} Throw away 1
Problem 2. 8779_{16} Throw away 1
Problem 3. F916**H** No throw away
Problem 4. 86EC**H** No throw away
Problem 5. $E00D_{16}$ No throw away
Problem 6. $0C63_{16}$ No throw away

Convert to IBM BCD Answers

Problem 1. Convert ASCII -736 to IBM Unpacked BCD.

0011	0111	0011	0011	1101	0110
Z	d	Z	d	S	d

Problem 2. Convert ASCII -736 to IBM Packed BCD.

0111	0011	0110	1101
d	d	d	S

Fixed Point Conversion Answers

Problem 1. 11 1011 0101.0011 01_{2} = 3B5.34**H**

Problem 2. $FED.CAB_{16}$ = 1111 1110 1101.1100 1010 1011**B**

Problem 3. $10\ 1100.011_{2}$ = 44.375

Problem 4. 107.625 = $110\ 1011.101_{2}$

Problem 5. $AD.E_{16}$ = 173.875

Problem 6. 430.75781 25 = 1AE.C2**H**

Partitioned Numbers

Problem A1. a. Left partition = 160
b. Right partition = 22
c. Sum = 182

Problem A2. a. Left partition = 96
b. Right partition = 12
c. Sum = 108

Subnet Answers

Problem B1. Convert the dotted decimal for IP address 17.10.127.5 to binary and hexadecimal, using the table below.

17								10								127								5							
31							24	23							16	15							8	7							0
0	0	0	1	0	0	0	1	0	0	0	0	1	0	1	0	0	1	1	1	1	1	1	1	0	0	0	0	0	1	0	1
1				1				0				A				7				F				0				5			

Problem B2. How many subnet addresses are possible in the subnet mask shown below?

31							24	23							16	15							8	7							0
0	0	0	0	1	0	1	0	x	x	x	x	x	x	x	x	x	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y

x = 1FF = 256 + 240 + 15 = 511. The number of subnet addresses is 512, where 0 is one of the possible addresses. (0 – 511)

Problem B3. How many host machine addresses are possible for each subnet in the subnet mask shown below?

31							24	23							16	15							8	7							0
0	0	0	0	1	0	1	0	x	x	x	x	x	x	x	x	x	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y

y = 7FFFH = 28672 + 3840 + 240 + 15 = 32767

The number of host machine addresses is 32768. Recall that 0 is one of the possible addresses. (0 – 32767). The number of host machines that can be on that subnet is two less than the number of possible addresses, or 32768 - 2 = 32766.

Problem B4. How many bits in the below mask need to be set aside to make sure each subnet can have at least 283 host machines?

31							24	23							16	15							8	7							0
0	0	0	0	1	0	1	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	y	y	y	y	y	y	y	y

9 bits are needed. 8 bits gives only 256 addresses (0 – 255). 9 bits gives 512 addresses, which is at least 283. 2^9 is the smallest power of two greater than or equal to 283.

Problem B5. How many subnet addresses are possible in the below mask if enough bits are set aside to make sure each subnet can have at least 283 host machines?

31							24	23							16	15							8	7							0
0	**0**	**0**	**0**	**1**	**0**	**1**	**0**	**x**	**x**	**x**	**x**	**x**	**x**	**x**	**x**	**x**	**x**	**x**	**x**	**x**	**x**	**x**	**y**	**y**	**y**	**y**	**y**	**y**	**y**	**y**	**y**

15 bits are available for subnet addresses if enough bits (9) are set aside for host machine addresses. This yields 32768 different addresses (0-32767).

Problem B6. What is the dotted decimal notation for the subnet mask needed for ensuring enough bits are set aside to ensure each subnet can have at least 283 host machines?

255								255								254								0							
31							24	23							16	15							8	7							0
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**x**	**x**	**x**	**x**	**x**	**x**	**x**	**x**	**x**	**x**	**x**	**x**	**x**	**x**	**x**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

The bits in the subnet mask coded with “1” and “x” are turned on. Bits coded “0” are turned off. The subnet mask in dotted decimal notation is 255.255.254.0 .

German Roman Numeral Answers

Problem 1. Write 390 B.C. in Roman numerals.

CCCXC Zero and negative numbers do not exist in Roman numerals. We attach B.C.

Another representation is XCD B.C.

500 is D.
400 is CD, which is 100 less than 500.
390 is XCD, which is 10 less than 400.

Ambiguity: This is ambiguous because Roman numerals are not associative.

XC is 90
(XC)D is 410

Only if you scan from right to left is your answer unique.

Problem 2. Write 407 A.D. in Roman numerals. CDVII A.D. We attach A.D.

500 is D.
400 is CD, which is 100 less than 500.
407 is CDVII, which is 7 more than 400.

Problem 3. Write 255 in Roman numerals. CCLV.

200 is CC.
250 is CCL.
255 is CCLV.

Problem 4. Add XI and XXXII. Group similar letters together, while maintaining the relative position of letters. Simplify by substituting aggregate symbols.

XXXX III → XLIII

The purpose of this problem is to illustrate that addition was fairly simple for the Romans.

Problem 5. XXV is 25. Time to get the clock fixed.

Problem 6. Zero cannot be represented by Roman numerals. It lacks nothing.

Problem 7. Divide XXX by VI.

I do not know of an easy way of doing division using Roman numerals.

Roman engineers used a calculating board or an abacus for doing arithmetic and translated the answers into Roman numerals.

One approach would be to expand both to all ones, and group the ones. Likewise, they needed some way of doing multiplication. You can see that calculation with Roman numerals is not as easy as calculation using our positional number system.

XXX = 30, VI = 6. 30/6 = 5 = V.

So, the Roman numeral system was good for enumeration, counting, and addition.

Problem 8. MDCCLXXVI Americans: You should recognize the answer after you decode it.

Problem 9. I(VL) = 54 – 1 = 44

Problem 10. (IV)L = 50 – 4 = 46

Fibonacci Bases Answers

Problem 1. Convert 50_{10} to Minimal Fibonacci Base (Zeckendorf) Representation.

50	50	50	50	50	50	16	16	3	3	3		
F_{14}	F_{13}	F_{12}	F_{11}	F_{10}	F_9	F_8	F_7	F_6	F_5	F_4	F_3	F_2
377	233	144	89	55	34	21	13	8	5	3	2	1
0	0	0	0	0	1	0	1	0	0	1	0	0
0	0	0	0	0	34	0	13	0	0	3	0	0
					16		3			0		

Remainder **R**
← Fibonacci Number
← place values
← digits
← decimal values
Remainder **R**

Problem 2. Convert 53_{10} to Minimal Fibonacci Base (Zeckendorf) Representation.

53	53	53	53	53	53	19	19	6	6	1	1	1
F_{14}	F_{13}	F_{12}	F_{11}	F_{10}	F_9	F_8	F_7	F_6	F_5	F_4	F_3	F_2
377	233	144	89	55	34	21	13	8	5	3	2	1
0	0	0	0	0	1	0	1	0	1	0	0	1
					34		13		5			1
					19		6		1			0

Remainder **R**
← Fibonacci Number
← place values
← digits
← decimal values
Remainder **R**

If you add the Zeckendorf Representation numbers for 50 + 53, you get:

F_{15}	F_{14}	F_{13}	F_{12}	F_{11}	F_{10}	F_9	F_8	F_7	F_6	F_5	F_4	F_3	F_2
610	377	233	144	89	55	34	21	13	8	5	3	2	1
						1	0	1	0	0	1	0	0
						1	0	1	0	1	0	0	1
						2	0	2	0	1	1	0	1
						68		26		5	3		1

← Fibonacci Number
← place values
← X
← Y
Answer
← decimal values

Converting this Fibonacci Base Representation to a Minimal Fibonacci Base Representation:

F_{14}	F_{13}	F_{12}	F_{11}	F_{10}	F_9	F_8	F_7	F_6	F_5	F_4	F_3	F_2	← Fibonacci Number
377	233	144	89	55	34	21	13	8	5	3	2	1	← place values
					2	0	2	0	1	1	0	1	Answer
					2	1	0	0	2	1	0	1	Rule 1
					2	1	0	1	0	1	1	1	Rule 1
				1	0	1	1	1	0	1	1	1	Rule 1
				1	0	1	1	1	0	2	0	0	Rule 2
				1	0	1	1	1	1	0	0	1	Rule 1
				1	0	1	2	0	0	0	0	1	Rule 2
				1	0	2	0	0	1	0	0	1	Rule 1
				1	1	0	0	1	1	0	0	1	Rule 1
				1	1	0	1	0	0	0	0	1	Rule 2
			1	0	0	0	1	0	0	0	0	1	Rule 2

The answer is $10\ 0010\ 0001_{Fib}$
To check the answer, this is $(1 \times 89) + (1 \times 13) + (1 \times 1) = 103_{10}$

Problem 3. Add the two Zeckendorf Representation numbers:
$1000\ 1010\ 0101_{Fib} + 1010\ 0001\ 0011_{Fib}$

F_{15}	F_{14}	F_{13}	F_{12}	F_{11}	F_{10}	F_9	F_8	F_7	F_6	F_5	F_4	F_3	F_2	← Fibonacci Number
610	377	233	144	89	55	34	21	13	8	5	3	2	1	← place values
		1	0	0	0	1	0	1	0	0	1	0	1	← X
		1	0	1	0	0	0	0	1	0	0	1	1	← Y
		2	0	1	0	1	0	1	1	0	1	1	2	Answer
		466		89		34		13	8		3	2	2	← decimal values

The sum is 617_{10}

Converting this Fibonacci Base Representation to a Minimal Fibonacci Base Representation:

F_{15}	F_{14}	F_{13}	F_{12}	F_{11}	F_{10}	F_9	F_8	F_7	F_6	F_5	F_4	F_3	F_2	← Fibonacci Number
610	**377**	**233**	**144**	**89**	**55**	**34**	**21**	**13**	**8**	**5**	**3**	**2**	**1**	← place values
		2	0	1	0	1	0	1	1	0	1	1	2	Answer
		2	0	1	0	1	0	1	1	0	1	2	0	Rule 1
		2	0	1	0	1	0	1	1	0	2	0	1	Rule 1
		2	0	1	0	1	0	1	1	1	0	0	2	Rule 1
		2	0	1	0	1	0	1	1	1	0	1	0	Rule 1
	1	0	0	2	0	1	0	1	1	1	0	1	0	Rule 1
	1	0	1	0	0	2	0	1	1	1	0	1	0	Rule 1
	1	0	1	0	1	0	0	2	1	1	0	1	0	Rule 1
	1	0	1	0	1	0	1	0	1	2	0	1	0	Rule 1
	1	0	1	0	1	0	1	0	2	0	0	2	0	Rule 1
	1	0	1	0	1	0	1	0	2	0	1	0	1	Rule 1
	1	0	1	0	1	0	1	1	0	0	2	0	1	Rule 1
	1	0	1	0	1	0	1	1	0	1	0	0	2	Rule 1
	1	0	1	0	1	0	1	1	0	1	0	1	0	Rule 1
	1	0	1	0	1	1	0	0	0	1	0	1	0	Rule 2
	1	0	1	1	0	0	0	0	0	1	0	1	0	Rule 2
	1	1	0	0	0	0	0	0	0	1	0	1	0	Rule 2
1	0	0	0	0	0	0	0	0	0	1	0	1	0	Rule 2

The answer is $10\ 0000\ 0000\ 1010_{Fib}$
Converted to decimal, it is: $(1 \times 610) + (1 \times 5) + (1 \times 2) = 617_{10}$

Problem 4. Restore Fibonacci Base Representation 10110112_{Fib} to Minimal Fibonacci Base Representation.

F_{14}	F_{13}	F_{12}	F_{11}	F_{10}	F_9	F_8	F_7	F_6	F_5	F_4	F_3	F_2	← Fibonacci Number
377	**233**	**144**	**89**	**55**	**34**	**21**	**13**	**8**	**5**	**3**	**2**	**1**	← place values
					1	0	1	1	0	1	1	2	Answer
					1	0	1	1	0	1	2	0	Rule 1
					1	0	1	1	0	2	0	1	Rule 1
					1	0	1	1	1	0	0	2	Rule 1
					1	0	1	1	1	0	1	0	Rule 1
					1	0	2	0	0	0	1	0	Rule 2
					1	1	0	0	1	0	1	0	Rule 1
				1	0	0	0	0	1	0	1	0	Rule 2

The answer is $1\ 0000\ 1010_{Fib}$
Converted to decimal, it is $(1 \times 55) + (1 \times 5) + (1 \times 2) = 62_{10}$

Phi Base Number System Answers

Problem 1. Express A = $\Phi^4 + \Phi^3 + \Phi^{-3}$ as phigits:

	F^6	F^5	F^4	F^3	F^2	F^1	F^0	.	F^{-1}	F^{-2}	F^{-3}	F^{-4}	F^{-5}	F^{-6}
A =			1	1	0	0	0	.	0	0	1			

Problem 2. Express B = $\Phi^4 + \Phi^1 + \Phi^{-4} + \Phi^{-6}$ as phigits:

	F^6	F^5	F^4	F^3	F^2	F^1	F^0	.	F^{-1}	F^{-2}	F^{-3}	F^{-4}	F^{-5}	F^{-6}
B =			1	0	0	1	0	.	0	0	0	1	0	1

Problem 3. Add the phigits A (from Problem 1) and B (from Problem 2).

	F^6	F^5	F^4	F^3	F^2	F^1	F^0	.	F^{-1}	F^{-2}	F^{-3}	F^{-4}	F^{-5}	F^{-6}
A =			1	1	0	0	0	.	0	0	1			
B =			1	0	0	1	0	.	0	0	0	1	0	1
A+B=			2	1	0	1	0	.	0	0	1	1	0	1

Problem 4. Transform the sum, A+B, computed in Problem 3 into minimal Phi base representation. In the left column, write the rule number being used for each step of the transformation.

	F^6	F^5	F^4	F^3	F^2	F^1	F^0	.	F^{-1}	F^{-2}	F^{-3}	F^{-4}	F^{-5}	F^{-6}
A+B=			2	1	0	1	0	.	0	0	1	1	0	1
R# 2		1	1	0	0	1	0	.	0	0	1	1	0	1
R# 2	1	0	0	0	0	1	0	.	0	0	1	1	0	1
R# 2	1	0	0	0	0	1	0	.		1	0	0	0	1

Problem 5. In the answer to Problem 4, substitute expressions for F^n in terms of n and f. Reduce the expression to the fewest number of terms.

$\Phi^6 + \Phi^1 + \Phi^{-2} + \Phi^{-6} = 13+8\varphi + 1+\varphi + 1-\varphi + 5-8\varphi = 20 + 0\varphi =$ **20**

Continued Fraction Answer

Problem 1. Evaluate the first six terms. Show all your steps.

$$X = 1+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\frac{1}{1}}}}} = 1+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\frac{1}{1+1}}}} = 1+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\frac{1}{2}}}} = 1+\cfrac{1}{1+\cfrac{1}{1+\frac{1}{1.5}}}$$

$$= 1+\cfrac{1}{1+\cfrac{1}{1.66\ldots}} = 1+\frac{1}{1.6} = 1.625$$

The 6-term continued fraction approximation to the golden ratio is 1.625. By comparison, the ten digit approximation of the golden ratio is 1.618033989.

Compute as follows:

(1) 1 Enter
(2) $x^{-1} + 1$
(3) Repeat step (2) four times

Complement Arithmetic Answers

Problem 1. 18
Problem 2. - 12
Problem 3. $20C_{16}$
Problem 4. - $03D_{16}$
Problem 5. $0001\ 1001_2$
Problem 6. - $1000\ 0111_2$

Fast Two's Complement Answers

Problem 1. Consider the binary number 00110100_2.

a. The right-most 1-bit is in bit position 2.
b. Leave bit positions 2, 1, and 0 alone.
c. The two's complement of this binary number is 11001100_2.

Problem 2. Consider the binary number 11010000_2.

a. The right-most 1-bit is in bit position 4.
b. Leave bit positions 4, 3, 2, 1, and 0 alone.
c. The two's complement of this binary number is 00110000_2.

Binary Multiplication Answer

Column Position Number						4	3	2	1	0
X						**1**	**1**	**1**	**0**	**1**
Y						**1**	**0**	**1**	**1**	**0**
Carry	1	1	1	1	1	0	0	0	0	
Column 0						0	0	0	0	0
Column 1					1	1	1	0	1	
Column 2				1	1	1	0	1		
Column 3			0	0	0	0	0			
Column 4		1	1	1	0	1				
Sum	**1**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**0**

Check the answer by converting $10\ 0111\ 1110_2$ to decimal.

Place Value	512	256	128	64	32	16	8	4	2	1
Sum	1	0	0	1	1	1	1	1	1	0

The answer is 638.

Hexadecimal Multiplication Answer

Problem 1. Multiply X = FAD**H** times EB_{16}.

Sum Carry	1		2		1					
Carry term 1			A		6		8			
Carry term 2	D		8		B					
X					**F**		**A**		**D**	
times Y							**E**		**B**	
					5		E		F	
			2		C		6			
Answer	E		6		3		C		F	

Answer = E63CF**H** = $953{,}055_{10}$.

Fixed Point Binary Division Answers

Problem 1. Divide 101.1100 11_2 by 11.0101_2 .

							1	.1	1
110101	1	0	1	1	1	0	0	.1	1
		1	1	0	1	0	1		
		1	0	0	1	1	1	1	
			1	1	0	1	0	1	
				1	1	0	1	0	1
				1	1	0	1	0	1
									0

The answer is 1.11_2

Prime Number Answers

Problem 1.
a. Show that 7 is a prime number using base 2.
b. Show that 7 is a prime number using base 3.
c. Compare the results.

a. Demonstrate 7 is a prime number using base 2.

Divide by 2

		1	1
10	1	1	1
	1	0	
		1	1
		1	0
		R_3=	1
		R_{10}	=1

3

		1	0
11	1	1	1
	1	1	
		0	1
			0
		R_3=	1
		R_{10}	1

5

			1
101	1	1	1
	1	0	1
	R_3=	1	0
		R_{10}	=2

Divide by 7

			1
111	1	1	1
	1	1	1
		R_3=	0
		R_{10}	=0

The first integer that evenly divides 111_2 is 111_2. Therefore, $111_2 = 7_{10}$ is a prime number.

b. Demonstrate that 7 is a prime number using base 3.

Divide by 2

	1	0
2	2	1
	2	
	0	1
		0
R_3=		1
R_{10}		=1

Divide by 3

		2
10	2	1
	2	0
R_3=		1
R_{10}		=1

Divide by 5

		1
12	2	1
	1	2
R_3=		2
R_{10}		=2

Divide by 7

		1
21	2	1
	2	1
R_3=		0
R_{10}		=0

The first integer that divides 21_3 evenly is 21_3. Therefore, $21_3 = 7_{10}$ is a prime number.

c. In both problems, the equivalent of 7_{10} is a prime number.

Scientific Notation Conversion Answers

Problem 1. 6.626196×10^{-34} J/s is Plank's constant.

Problem 2. 2.9979250×10^{8} m/s is the speed of light in a vacuum.

Scientific Notation Multiplication Answers

Problem 1. Approximate the distance of one light-year:

$(2.998 \times 10^{8}\ \text{m/s}) \times (3.147 \times 10^{7}\ \text{s/yr}) \times (1\ \text{yr}) = 9.435 \times 10^{15}\ \text{m}$
$= 9.435 \times 10^{12}\ \text{km}$

Problem 2. Naively estimate the kinetic energy of the earth revolving around the Sun, assuming a circular orbit (ignoring Kepler):

$(1/2) \times (5.98 \times 10^{24}\ \text{kg}) \times (2.99 \times 10^{4}\ \text{m/s})^2 = 2.67 \times 10^{33}\ \text{J}$

Floating Point Conversion Answers

Problem 1.

IEEE Single Precision Floating Point

S	Characteristic								Mantissa						
0	1			4				8	9			12			15
0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0

Mantissa (Continued)															
16				20				24				28			31
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Procedure:

Step 1: Let M = (IEEE Single Precision Floating Point characteristic) – (IEEE Single Precision Floating Point Excess). M = 129 – 127 = 2.

Note: The IEEE format uses implicit normalization. The radix point is immediately to the left of the IEEE mantissa most significant bit.

Step 2: Insert the prefix bit pattern 0001 immediately to the left of the radix point. 0001.100 0000 0000 0000 0000 0001

Step 3: Let MM = M MOD 4. This is the remainder obtained by dividing M by 4. MM = 2 MOD 4 = 2.

Step 4: Compute the number of bits (J) to right-shift the mantissa and prefix bit pattern.

- If MM = 0, then J = 4.
- If MM < 0, then J = |MM|. |MM| is the absolute value of MM.
- If MM > 0, then J = 4 – MM. J = 4 – 2 = 2.

The number of bits of precision in the mantissa lost is J – 1 = 2 – 1 = 1.

Step 5: Copy the shifted bit pattern into the IBM Short HFP mantissa.

Step 6: Compute the IBM HFP characteristic, without excess.

- $K = \left\lfloor \frac{M+4}{4} \right\rfloor = \left\lfloor \frac{2+4}{4} \right\rfloor = \left\lfloor \frac{6}{4} \right\rfloor = \lfloor 1.5 \rfloor = 1.$

Step 7: Add the IBM HFP excess to K and store it in the HFP characteristic field. K + 64 = 65 = 41_{16} = $100\ 0001_2$.

Step 8: Copy the sign bit from the IEEE Single Precision FP word to the sign bit of the IBM HFP word.

IBM Hexadecimal Short Floating Point

S	Characteristic							Mantissa							
0	1			4			7	8				12			15
0	1	0	0	0	0	0	1	0	1	1	0	0	0	0	0

Mantissa (Continued)															
16				20				24				28			31
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Problem 2.

IEEE Single Precision Floating Point

S	Characteristic								Mantissa						
0	1			4				8	9			12			15
0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	0

Mantissa (Continued)															
16				20				24				28			31
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Procedure:

Step 1: Let M = (IEEE Single Precision Floating Point characteristic) – (IEEE Single Precision Floating Point Excess). M = 132 – 127 = 5.

Note: The IEEE format uses implicit normalization. The radix point is immediately to the left of the IEEE mantissa most significant bit.

Step 2: Insert the prefix bit pattern 0001 immediately to the left of the radix point. 0001.010 0000 0000 0000 0000 0000

Step 3: Let MM = M MOD 4. This is the remainder obtained by dividing M by 4. MM = 5 MOD 4 = 1.

Step 4: Compute the number of bits (J) to right-shift the mantissa and prefix bit pattern.

- If MM = 0, then J = 4
- If MM < 0, then J = |MM|. |MM| is the absolute value of MM.
- If MM > 0, then J = 4 – MM = 4 – 1 = 3.

The number of bits of precision in the mantissa lost is J – 1 = 3 – 1 = 2.

Step 5: Copy the shifted bit pattern into the IBM Short HFP mantissa.

Step 6: Compute the IBM HFP characteristic, without excess.

- $K = \left\lfloor \frac{M+4}{4} \right\rfloor = \left\lfloor \frac{5+4}{4} \right\rfloor = \left\lfloor \frac{9}{4} \right\rfloor = \lfloor 2.25 \rfloor = 2$.

Step 7: Add the IBM HFP excess to K and store it in the HFP characteristic field. K + 64 = 2 + 64 = 66 = 42_{16} = $100\ 0010_2$.

Step 8: Copy the sign bit from the IEEE Single Precision FP word to the sign bit of the IBM HFP word.

IBM Hexadecimal Short Floating Point

S	Characteristic							Mantissa							
0	1			4			7	8				12			15
0	1	0	0	0	0	1	0	0	0	1	0	1	0	0	0

Mantissa (Continued)															
16				20				24				28			31
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Problem 3.

IEEE Single Precision Floating Point

S	Characteristic								Mantissa						
0	1			4				8	9			12			15
1	0	1	1	1	1	1	0	1	0	0	1	0	0	0	0

Mantissa (Continued)															
16				20				24				28			31
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Procedure:

Step 1: Let M = (IEEE Single Precision Floating Point characteristic) – (IEEE Single Precision Floating Point Excess). M = 125 – 127 = - 2 .

Note: The IEEE format uses implicit normalization. The radix point is immediately to the left of the IEEE mantissa most significant bit.

Step 2: Insert the prefix bit pattern 0001 immediately to the left of the radix point. 0001.001 0000 0000 0000 0000 0000

Step 3: Let MM = M MOD 4. This is the remainder obtained by dividing M by 4. MM = -2 MOD 4 = -2 .

Step 4: Compute the number of bits (J) to right-shift the mantissa and prefix bit pattern.

- If MM = 0, then J = 4
- If MM < 0, then J = |MM|. J = | -2 | = 2 .
- If MM > 0, then J = 4 – MM.

The number of bits of precision in the mantissa lost is J – 1 = 2 – 1 = 1.

Step 5: Copy the shifted bit pattern into the IBM Short HFP mantissa.

Step 6: Compute the IBM HFP characteristic, without excess.

- $K = \left\lfloor \frac{M+4}{4} \right\rfloor = \left\lfloor \frac{-2+4}{4} \right\rfloor = \left\lfloor \frac{2}{4} \right\rfloor = \lfloor 0.5 \rfloor = 0$.

Step 7: Add the IBM HFP excess to K and store it in the HFP characteristic field. K + 64 = 0 + 64 = 64 = 40_{16} = $100\ 0000_2$.

Step 8: Copy the sign bit from the IEEE Single Precision FP word to the sign bit of the IBM HFP word.

IBM Hexadecimal Short Floating Point

S	Characteristic							Mantissa							
0	1			4			7	8			12				15
1	1	0	0	0	0	0	0	0	1	0	0	1	0	0	0

Mantissa (Continued)															
16				20				24				28			31
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Problem 4.

IBM Hexadecimal Extended Floating Point

High Order Word

S	High Order Characteristic							High Order Mantissa							
0	1			4			7	8				12			15
0	1	0	0	1	0	1	0	0	0	0	1	0	1	1	0

High Order Mantissa (Continued 1)															
16				20				24				28			31
0	1	0	1	1	1	0	0	1	1	1	1	1	0	0	1

High Order Mantissa (Continued 2)															
32				36				40				44			47
0	0	0	1	0	0	0	0	0	1	0	1	1	0	0	1

High Order Mantissa (Continued 3)															
48				52				56				60			63
1	1	1	0	1	1	1	0	0	1	0	1	0	0	1	0

Low Order Word

S	Low Order Characteristic							Low Order Mantissa							
64	65			68			71	72				76			79
0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0

Low Order Mantissa (Continued 1)															
80				84				88				92			95
0	0	0	0	1	0	1	1	0	1	1	1	0	0	1	0

Low Order Mantissa (Continued 2)															
96				100				104				108			111
1	1	0	0	0	1	0	1	1	0	0	0	1	0	1	0

Low Order Mantissa (Continued 3)															
112				116				120				124			127
1	1	1	0	0	1	0	0	0	1	0	0	0	1	0	1

Procedure:

Step 1: Subtract the IBM Hexadecimal Floating Point Excess from the high order characteristic.

Step 2: Let K = the numerical value of the high order characteristic. K = 74 – 64 = 10

Step 3: Let N = 4 K. Each positive increment of the IBM Hexadecimal Floating Point characteristic represents a shift of the radix by 4 bit positions to the left to perform the normalization. $N = 4 \times 10 = 40$.

Step 4: Let J = the number of bit positions from the left end of the mantissa occupied by the first one-bit. $J = 4$.

Step 5: Let $P = N - J = 40 - 4 = 36$.

Step 6: Add the IEEE Double Extended Precision Floating Point Excess to P. $M = P + XS_{IEEE\ DEP\ FP} = 36 + 16383 = 16419$.

Step 7: Convert M to binary. $16419 = 4023_{16} = 100\ 0000\ 0010\ 0011_2$.

Step 8: Record the result in the characteristic field of the IEEE Double Extended Precision Floating Point word.

Step 9: The most significant bit of the mantissa of the IBM Hexadecimal Floating Point becomes the implicit bit in the IEEE Double Extended Precision Floating Point word, and therefore does not explicitly get recorded.

Step 10: Beginning with the bit immediately to the right of the most significant bit, copy the remaining bits from the IBM Hexadecimal Floating Point mantissa into the IEEE Double Extended Precision Floating Point mantissa.

Step 11: Stop after the IEEE Double Extended Precision Floating Point mantissa is filled. Note that 44 to 47 least-significant-bits of precision in the mantissa were lost in the conversion. The number of bits of precision lost is $48 - J = 48 - 4 = 44$.

IEEE Double Extended Precision Floating Point

S	Characteristic														
79	78			75			72	71				67			64
0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	1

Mantissa															
63			60				56				52				48
0	1	1	0	0	1	0	1	1	1	0	0	1	1	1	1

Mantissa (Continuation 1)															
47			44				40				36				32
1	0	0	1	0	0	0	1	0	0	0	0	0	1	0	1

Mantissa (Continuation 2)															
31			28				24				20				16
1	0	0	1	1	1	1	0	1	1	1	0	0	1	0	1

Mantissa (Continuation 3)															
15			12				8				4				0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Problem 5. Convert the following IEEE Double Extended Precision Floating Point word to an IBM Hexadecimal Extended Floating Point word.

IEEE Double Extended Precision Floating Point

S	Characteristic														
79	78			75			72	71				67			64
1	0	1	1	1	1	1	1	0	1	0	0	1	0	1	0

Mantissa															
63			60				56				52				48
0	1	1	1	0	0	1	0	1	1	1	1	1	0	1	1

Mantissa (Continuation 1)															
47			44				40				36				32
1	1	0	0	0	1	1	1	0	0	1	1	1	1	0	0

Mantissa (Continuation 2)															
31			28				24				20				16
1	1	1	1	1	0	0	0	0	1	1	0	1	1	0	0

Mantissa (Continuation 3)															
15			12				8				4				0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

Procedure:

Step 1: Set all IBM Hexadecimal Extended Floating Point bits to zero.

Step 2: Copy the IEEE Double Extended Precision Floating Point sign bit into the IBM Hexadecimal Extended Floating Point High Order sign bit and Low Order sign bit.

Step 3: Evaluate the IEEE Double Extended Precision Floating Point Excess from the characteristic. $3F4A_{16} = (3 \times \mathbf{16^3}) + (F \times \mathbf{16^2}) + (4 \times \mathbf{16^1}) + (A \times \mathbf{16^0}) = 12288 + 3840 + 64 + 10 = 16202$.

Step 4: Subtract the IEEE Double Extended Precision Floating Point Excess from the characteristic. $P = 16202 - 16383 = -181$.

Step 5: Compute the IBM Hexadecimal Floating Point Characteristic without the IBM HFP Excess. $K = \left\lfloor \frac{P+4}{4} \right\rfloor = \left\lfloor \frac{-181+4}{4} \right\rfloor = \left\lfloor \frac{-177}{4} \right\rfloor = \lfloor -44.25 \rfloor = -45$

Step 6: Compute the IBM Hexadecimal Floating Point Characteristic with the IBM HFP Excess. $N = K + 64 = -45 + 64 = 19 = 13_{16} = 001\ 0011_2$ ..

Step 7: Write N into the IBM HFP High Order Characteristic.

Step 8: Append the implicit most significant bit to the left of the radix point for the IEEE DEP FP mantissa.

Step 9: Let MM = P MOD 4. This is the remainder obtained by dividing P by 4. MM = -181 MOD 4 = -1 .

Step 10: Compute the number of places to shift the appended mantissa to obtain the IBM HFP mantissa.

- If MM = 0, then J = 4
- If MM > 0, then J = 4 – MM
- If MM < 0, then J = | MM | where the vertical bars identify the absolute value function.
  J = | -1 | = 1

Step 11: Beginning with the most significant bit of the shifted mantissa, copy the remaining bits from the IEEE Double Extended Precision Floating Point mantissa into the IBM Hexadecimal Floating Point Mantissa.

Step 12: Compute the IBM HFP Low Order Characteristic.
Q = N – 14 = 19 – 14 = 5 .

Step 13: Write Q into the IBM HFP Low Order Characteristic.

IBM Hexadecimal Extended Floating Point

High Order Word

S	High Order Characteristic							High Order Mantissa							
0	1			4			7	8			12				15
1	0	0	1	0	0	1	1	1	0	1	1	1	0	0	1

High Order Mantissa (Continued 1)															
16				20				24				28			31
0	1	1	1	1	1	0	1	1	1	1	0	0	0	1	1

High Order Mantissa (Continued 2)															
32				36				40				44			47
1	0	0	1	1	1	1	0	0	1	1	1	1	1	0	0

High Order Mantissa (Continued 3)															
48				52				56				60			63
0	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0

Low Order Word

S	Low Order Characteristic							Low Order Mantissa							
64	65			68			71	72			76				79
1	0	0	0	0	1	0	1	0	0	0	0	1	0	0	0

Low Order Mantissa (Continued 1)															
80				84				88				92			95
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Low Order Mantissa (Continued 2)															
96				100				104				108			111
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Low Order Mantissa (Continued 3)															
112				116				120				124			127
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits 81 through 111 are all default values of zero.

Problem 6.

IEEE Double Extended Precision Floating Point

S	Characteristic														
79	78			75			72	71				67			64
1	0	1	1	1	1	1	1	0	0	0	0	1	0	1	1

Mantissa															
63			60				56				52				48
0	1	1	1	0	0	1	0	1	1	1	1	1	0	1	1

Mantissa (Continuation 1)															
47			44				40				36				32
1	1	0	0	0	1	1	1	0	0	1	1	1	1	0	0

Mantissa (Continuation 2)															
31			28				24				20				16
1	1	1	1	1	0	0	0	0	1	1	0	1	1	0	0

Mantissa (Continuation 3)															
15			12				8				4				0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

Procedure:

Step 1: Set all IBM Hexadecimal Extended Floating Point bits to zero.

Step 2: Copy the IEEE Double Extended Precision Floating Point sign bit into the IBM Hexadecimal Extended Floating Point High Order sign bit and Low Order sign bit.

Step 3: Evaluate the IEEE Double Extended Precision Floating Point Excess from the characteristic. $3F0B_{16} = (3 \times \mathbf{16^3}) + (F \times \mathbf{16^2}) + (0 \times \mathbf{16^1}) + (B \times \mathbf{16^0}) = 12288 + 3840 + 0 + 11 = 16139$.

Step 4: Subtract the IEEE Double Extended Precision Floating Point Excess from the characteristic. $P = 16139 - 16383 = -244$.

Step 5: Compute the IBM Hexadecimal Floating Point Characteristic without the IBM HFP Excess. $K = \left\lfloor \frac{P+4}{4} \right\rfloor = \left\lfloor \frac{-244+4}{4} \right\rfloor = \left\lfloor \frac{-240}{4} \right\rfloor = -60$

Step 6: Compute the IBM Hexadecimal Floating Point Characteristic with the IBM HFP Excess. N = K + 64 = -60 + 64 = 4 .

Step 7: Write N into the IBM HFP High Order Characteristic.

Step 8: Append the implicit most significant bit to the left of the radix point for the IEEE DEP FP mantissa.

Step 9: Let MM = P MOD 4. This is the remainder obtained by dividing P by 4. MM = -244 MOD 4 = 0 .

Step 10: Compute the number of places to shift the appended mantissa to obtain the IBM HFP mantissa.

- If MM = 0, then J = 4
- If MM > 0, then J = 4 – MM
- If MM < 0, then J = | MM | where the vertical bars identify the absolute value function.

  J = 4

Step 11: Beginning with the most significant bit of the shifted mantissa, copy the remaining bits from the IEEE Double Extended Precision Floating Point mantissa into the IBM Hexadecimal Floating Point High Order Mantissa.

Step 12: Compute the IBM HFP Low Order Characteristic.
Q = N – 14 = 4 – 14 = -10.

We cannot proceed further because –10 is not a possible value for an IBM HFP Characteristic. Note that we can approximate the IEEE word by using an IBM Hexadecimal Long Floating Point word which does not need to use a low order characteristic.

IBM Hexadecimal Extended Floating Point

High Order Word

S	High Order Characteristic							High Order Mantissa							
0	1			4			7	8			12				15
1	0	0	0	0	1	0	0	0	0	0	1	0	1	1	1

High Order Mantissa (Continued 1)															
16				20				24				28			31
0	0	1	0	1	1	1	1	1	0	1	1	1	1	0	0

High Order Mantissa (Continued 2)															
32				36				40				44			47
0	1	1	1	0	0	1	1	1	1	0	0	1	1	1	1

High Order Mantissa (Continued 3)															
48				52				56				60			63
1	0	0	0	0	1	1	0	1	1	0	0	0	0	0	0

Low Order Word

S	Low Order Characteristic							Low Order Mantissa							
64	65			68			71	72			76				79
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Low Order Mantissa (Continued 1)															
80				84				88				92			95
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Low Order Mantissa (Continued 2)															
96				100				104				108			111
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Low Order Mantissa (Continued 3)															
112				116				120				124			127
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Problem 7.

Convert the following IEEE Double Extended Precision Floating Point word to an IBM Hexadecimal Extended Floating Point word.

IEEE Double Extended Precision Floating Point

S	Characteristic														
79	78			75			72	71				67			64
1	0	1	1	1	0	1	1	1	0	0	1	0	1	1	0

Mantissa															
63			60				56				52				48
0	1	1	1	0	0	1	0	1	1	1	1	1	0	1	1

Mantissa (Continuation 1)															
47			44				40				36				32
1	1	0	0	0	1	1	1	0	0	1	1	1	1	0	0

Mantissa (Continuation 2)															
31			28				24				20				16
1	1	1	1	1	0	0	0	0	1	1	0	1	1	0	0

Mantissa (Continuation 3)															
15			12				8				4				0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

Procedure:

Step 1: Set all IBM Hexadecimal Extended Floating Point bits to zero.

Step 2: Copy the IEEE Double Extended Precision Floating Point sign bit into the IBM Hexadecimal Extended Floating Point High Order sign bit and Low Order sign bit.

Step 3: Evaluate the IEEE Double Extended Precision Floating Point Excess from the characteristic. $3B96_{16} = (3 \times \mathbf{16^3}) + (B \times \mathbf{16^2}) + (9 \times \mathbf{16^1}) + (6 \times \mathbf{16^0}) = 12288 + 2816 + 144 + 6 = 15254$.

Step 4: Subtract the IEEE Double Extended Precision Floating Point Excess from the characteristic. $P = 15254 - 16383 = -1129$.

Step 5: Compute the IBM Hexadecimal Floating Point Characteristic without the IBM HFP Excess. $K = \left\lfloor \frac{P+4}{4} \right\rfloor = \left\lfloor \frac{-1129+4}{4} \right\rfloor = \left\lfloor \frac{-1125}{4} \right\rfloor = \lfloor -281.25 \rfloor = -282$

We cannot proceed further because –282 is not a possible value for an IBM HFP Characteristic.

IBM Hexadecimal Extended Floating Point

High Order Word

S	High Order Characteristic							High Order Mantissa							
0	1			4			7	8			12				15
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

High Order Mantissa (Continued 1)															
16				20				24				28			31
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

High Order Mantissa (Continued 2)															
32				36				40				44			47
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

High Order Mantissa (Continued 3)															
48				52				56				60			63
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Low Order Word

S	Low Order Characteristic							Low Order Mantissa							
64	65			68			71	72			76				79
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Low Order Mantissa (Continued 1)															
80				84				88				92			95
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Low Order Mantissa (Continued 2)															
96				100				104				108			111
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Low Order Mantissa (Continued 3)															
112				116				120				124			127
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0