CS/ECE 552 : Introduction to Computer Architecture
Spring 2010
Prof. Wood
Problem Set #6 Solutions

Problem 1:

```
Code:           Data bits
                 7 6 5 4 3 2 1 0
Check bit 1      0 1 0 1 1 0 1 1
Check bit 2      0 1 1 0 1 1 0 1
Check bit 3      1 0 0 0 1 1 1 0
Check bit 4      1 1 1 1 0 0 0 0
```

Part1:

Data bits – check bits – code word

```
1) 00000010 - 0101 - 010100000010
2) 10101000 - 0001 - 101110101000
3) 01100110 - 0010 - 001001100110
4) 11011011 - 1111 – 111111011011
```

Part 2:

```
1)            checkbits =>syndrome - correct data word
1) 0000.00000111 - 0000 => 0000 - 00000111  No error
2) 1011.10000111 - 1100 => 0111 - 10001111  name 3
3) 0010.10101001 - 0010 => 0000 - 10101001 No error
4) 0111.11001010 - 0101 => 0010 - 11001010  name 1
```

Problem 2:

```
       11010101  <- multiplicand (-43)
       01010101  <- multiplier   (85)
        1 1 1 1  <- booth encoding of multiplier
 ---------------
 11111111010101
 111111010101__
 1111010101____
 11010101______
 ---------------
 11000110111001  (-3655)
```

Problem 3: (109/6 = 18 rem 1)

Iteration	Step	Quotient	Divisor	Remainder
0	Initial Values	00000	01100000	01101101
1	R = R – D	00000	01100000	00001101
	R > 0, sll Q, Q0 = 1	00001	01100000	00001101
	Shift Div Right	00001	00110000	00001101
2	R = R – D	00001	00110000	11011101
	R < 0, sll Q, Q0 = 0	00010	00110000	00001101
	Shift Div, Revert Rem	00010	00011000	00001101
3	R = R – D	00010	00011000	11110101
	R < 0, sll Q, Q0 = 0	00100	00011000	00001101
	Shift Div, Revert Rem	00100	00001100	00001101
4	R = R – D	00100	00001100	00000001
	R > 0, sll Q, Q0 = 1	01001	00001100	00000001
	Shift Div Right	01001	00000110	00000001
5	R = R – D	01001	00000110	**11111011**
	R < 0, sll Q, Q0 = 0	**10010**	00000110	00000001
	Shift Div Right	10010	00000011	00000001

Problem 4.1:

343.723
343 = 101010111

.723 * 2 = **1**.446
.446 * 2 = **0**.892
.892 * 2 = **1**.784
.784 * 2 = **1**.568
.568 * 2 = **1**.136
.136 * 2 = **0**.272
.272 * 2 = **0**.544
.088 * 2 = **1**.088
etc...

343.723 = 101010111.101110010001011:0100001
Normalize => 1.01010111101110010001011:010* 2^7 (round down)
Mantissa => 01010111101110010001011
=> e=8 → E=e + Bias = +127 = 135
E = 10000111
Sign = 0
Representation: 0.10000111.01010111101110010001011 (0x43ABDC8B)

Problem 4.2:

1100 0000 1100 0100 1001 0000 0000 0000
Sign = 1
E = 10000001
Mantissa = 100 0100 1001 0000 0000 0000

E = 10000001 => e = E - Bias = 128 -127 = 2

F = 1.10001001001000000000000 * 2^2

= 110.00100100100000000000 = 2^-3 + 2 ^-6 + 2 ^ -9

= -6.142578125

Problem 5:

Initial state:

P0		P1		P2		Memory
State	**Data**	**State**	**Data**	**State**	**Data**	**Data**
I	**0**	**I**	**0**	**I**	**0**	**7**

Operation: Processor P0 executes lw $1, 100($0)

Action: P0 misses in the cache and reads the block from memory

Memory state:

P0		P1		P2		Memory
State	**Data**	**State**	**Data**	**State**	**Data**	**Data**
S	**7**	**I**	**0**	**I**	**0**	**7**

Operation: Processor P1 executes addi $1, $0, 13 and sw $1, 100($0)

Action: P1 misses, reads the block from memory and invalidates P0

Memory state:

P0		P1		P2		Memory
State	**Data**	**State**	**Data**	**State**	**Data**	**Data**
I	**7**	**M**	**13**	**I**	**0**	**7**

Operation: Processor P2 executes lw $1, 100($0)

Action: P2 misses, reads the block, P1 intervenes and supplies the block, updating memory

Memory state:

P0		P1		P2		Memory
State	**Data**	**State**	**Data**	**State**	**Data**	**Data**
I	**7**	**S**	**13**	**S**	**13**	**13**

Operation: Processor P1 executes addi $1, $0, 19 and sw $1, 100($0)

Action: P1 hits, but does not have permission to write. P1 invalidates P2's copy.

Memory state:

P0		P1		P2		Memory
State	**Data**	**State**	**Data**	**State**	**Data**	**Data**
I	**7**	**M**	**19**	**I**	**13**	**13**