HW6 Solutions (CS552 Spring 2013)

Problems 2

1.1

Data bits – check bits – code word

1) 00000111- 0000 – 0000 0000 0111
2) 10001111- 1011 – 1011 1000 1111
3) 11001010- 0101 – 0101 1100 1010
4) 10101001- 0010 – 0010 1010 1001

1.2

check bits – xor syndrome correct data word

1) 001111111111- 0011 – 0011 => 0000 - 11111111
   0011 No error
2) 001101111111- 1111 – 1111 => 1100 - 11111111
   0011 name 7
3) 111000011111- 1110 – 1110 => 0000 - 00011111
   1110 No Error
4) 011110001011- 1101 – 1101 => 1010 - 10101011
   0111 name 5

Problem 3

Part 1:

17 → 10001

0.384 < 2^-1 0

0.384 < 2^-2 1

0.384 < 2^-3 1

0.384 < 2^-4 0 → 0.0110001001001101111

0.384 < 2^-5 0

0.384 < 2^-6 0

0.384 < 2^-7 1

etc ...

17.384 => 10001.0110001001001101111

=> Normalize 1: 1.00010110001001001101111 x 2^4

=> e =4 → E =e + Bias = 4+127 = 131

E = 10000011 sign = 0, exponent = 10000011, fraction = 00010110001001001101111 Floating point representation :

01000001100010110001001001101111

Part 2:
0 00001000 000 0100 1001 0000 1001 0110 sign
= 0
E = 00001000 ==> e = E – Bias = 8 -127 = -119
F = 00001001001000010010110 ==> F = 2^-5 + 2^-8 + 2^-11+2^-16+2^-19+2^-21+2^-22
F = 1.03566241264343
Final result : 1.03566241264343 x 2^-119 = **1.5582916 x (10^-36)**

Problem 4

1) 32bits, 30bits
2) 2^20 entries
3) A tag identifies a unique virtual page number. Since there are 4G/4K (ie, 2^20) virtual pages, the tags should be 20 bits wide.
   Since there are 1G/4K (ie, 2^18) physical pages, the physical page numbers are 18 bits wide.

Problem 5

32'h0000_7000	TLB hit, 32'h0000_4000
32'h0000_B000	TLB hit, 32'h0000_C000
32'h0000_3000	TLB hit, 32'h0000_6000
32'h0000_BA5A	TLB hit, 32'h0000_CA5A
32'h0000_BFFF	TLB hit, 32'h0000_CFFF
32'h0000_AA5A	TLB miss, Page table hit, 32'h0000_3A5A
32'h0000_5FFF	TLB miss, Page table hit, 32'h0000_BFFF
32'h0000_1CCC	TLB miss, Page fault

Problem 6

A program with a lot of data dependances will do very badly in a GPU.
Eg,

```
int         A[100000];
//<initialize A here>;
int i;

for(i=2; i<100000)
{
        A[i] = A[i-1] + A[i-2];
}
```

In the above program, each iteration in the loop is dependent on the data from the previous two iterations. So, even if we were to split this workload across many threads in a GPU, the threads will have to synchronize and perform these operations sequentially.