Homework #1 **M1522.000800 System Programming**

Name: _____

Due Date: Tuesday, March 10, 2015, 23:59 **Student-Number:** _____

Submission: in paper form.
There will be a drop off box in class and in front of the CSAP Lab in building 301, room 419.

Question 1

Number Formats (Hexadecimal and Binary notations)

Perform the following number conversions and calculations.

example) `0x2015` in binary notation ('0x' represents hexadecimal number.)

=> 0b10000000010101 ('0b' represents binary number.)

a) 0b 1101 1110 1010 1101 1011 1110 1110 1111 in hexadecimal notation

b) `0xFACE` in binary notation

c) Your Student-Number (encoded in decimal numbers) in hexadecimal notation

Question 2

Number Formats (Hexadecimal and Binary notations)

Fill in the missing entries in the following figure, giving the decimal, binary, and hexadecimal values of different byte patterns.

Decimal (0d)	Binary (0b)	Hexadecimal (0x)
0	0000 0000	00
		ACE
191		
	1011 1010 1011 1110	

Question 3

Number Formats (Two's Complement Binary Representations)

Fill in the missing entries in the following figure, giving the binary, and decimal values of different patterns. (Use Two's Complement as representing signed numbers, and all formats follow 8-bit.)

Decimal (0d)	Binary (0b)	Hexadecimal (0x)
- 34	1101 1110	DE
	1000 0000	
		FF
- 84		

Question 4

Computer Architectures (Pipeline)

Explain why pipeline architectures are faster than sequential architectures in instruction-level parallelism.

Question 5

Assembly Language Programming

Assume the following values are stored at the indicated memory addresses and registers:

Address	Value
0x1000	0x0000_0000
0x1004	0x0000_0000

Register	Value
%eax	0x0000_1000
%ecx	0x0000_0001
%edx	0x0000_0003

Fill in the following table showing the effects of the following instructions, both in terms of the register or memory location that will be updated and the resulting value. (All instructions follow AT&T syntax, and might affect the next instructions.)

Instruction	Destination	Value
`movl %eax, (%eax)`		
`addl %ecx, %edx`		
`movl %eax, (%eax, %edx, 1)`		
`leal (%eax, %ecx, 4), %eax`		
`addl %edx, (%eax)`		

Question 6

Computer Architectures (Code Optimizations)

Consider the two C functions `sum1()` and `sum2()` below. Both compute the same result (the sum of all eight parameters), both use 7 addition operations to do so. Which one runs faster and why?

```
int sum1(int a,int b,int c,int d,
         int e,int f,int g,int h)
{
   int s = a;
   s += b;
   s += c;
   s += d;
   s += e;
   s += f;
   s += g;
   s += h;

   return s;
}
```

```
int sum2(int a,int b,int c,int d,
         int e,int f,int g,int h)
{
   int s1 = a+b;
   int s2 = c+d;
   int s3 = e+f;
   int s4 = g+h;

   s1 = s1 + s2;
   s3 = s3 + s4;
   s1 = s1 + s3;

   return s1;
}
```