Machine-Level Representation of Programs #1	**4190.308 Computer Architecture**

Due Date:	Wednesday, September 11, 2014, 23:59	**Solution**
Submission:	**Please update your e-mail address and mobile phone number on the eTL** in paper form. There will be a drop off box in class and in front of the CSAP Lab in building 301, room 419.	

Assignment

Update your information on eTL. (your picture, e-mail address and mobile phone number)

Question 1

Memory operations

Explain the difference of “`reg` to `mem`”, “`mem` to `reg`” and “`mem` to `mem`” ISAs.

- We have learned about the operand combination of Register, and Memory. Register is one of 8 integer registers. Otherwise, memory is 4 consecutive bytes of memory at address. So it works in “ address mode”. Through this definition, we can load the value from memory to register or store the value from register to memory in single instruction i.e movl. But in “mem-to-mem” case, we cannot access the the value from two memory addresses at once. It means that we cannot do that through single instruction. That’s the difference between 2 operations.

Question 2
Operand Specifiers

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

Address	**Value**		**Register**	**Value**
`0x100`	`0xFF`		`%eax`	`0x104`
`0x104`	`0xAB`		`%ecx`	`0x1`
`0x108`	`0x13`		`%edx`	`0x3`
`0x10C`	`0x11`			
`0x110`	`0xB5`			
`0x114`	`0x3B`			
`0x118`	`0x1A`			
`0x11C`	`0x03`			

Fill in the following table showing the values for the indicated operands:

Operand	**Value**
`%edx`	`0x3`
`0x114`	`0x3B`
`$0x11C`	`0x11C`
`(%eax)`	`0xAB`
`8(%eax)`	`0x11`
`0xD(%eax,%edx)`	`0x3B`
`264(%ecx,%edx)`	`0x11`
`0xFC(,%ecx,8)`	`0xAB`
`(%eax,%edx,8)`	`0x03`

Question 3

Data movement instructions

You are given the following information. A function with prototype

```
void decode(int *xp, int *yp, int *zp);
```

is compiled into assembly code. The body of the code is as follows:

`xp` at `%ebp+0x8`, `yp` at `%ebp+0xc`, `zp` at `%ebp+0x10`

```
mov    0x8(%ebp),%edx
mov    0xc(%ebp),%eax
mov    0x10(%ebp),%ecx
mov    (%eax),%esi
mov    (%ecx),%ebx
mov    (%edx),%edi
mov    %edi,(%ecx)
mov    %esi,(%edx)
mov    %ebx,(%eax)
```

Parameters `xp`, `yp`, and `zp` are stored at memory locations with offsets `0x8`, `0xc`, and `0x10`, respectively, relative to the address in register `%ebp`.
Write C code for `decode` that will have an effect equivalent to the assembly code above.

```
void decode2(int *xp, int *yp, int *zp)
{
   int tx = *xp;
   int ty = *yp;
   int tz = *zp;

   *zp = tx;
   *xp = ty;
   *yp = tz;
}
```