ECGR 4101/5101 - Assignment 1

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

Address	Value
0x100	0xFF
0x104	0xAB
0x108	0x13
0x10C	0x11

Register	Value
%eax	0x100
%ecx	0x1
%edx	0x3

For the following, determine the values for the indicated operands-

A. %eax
B. 0x104
C. $0x108
D. (%eax)
E. 4(%eax)
F. 9(%eax,%edx)
G. 260(%ecx,%edx)
H. 0xFC(,%ecx,4)
I. (%eax,%edx,4)

A: 0x100
B: 0xAB
C: 0x108
D: 0xFF
E: 0xAB
F: 0x11
G: 0x13
H: 0xFF
I: 0x11

2. You are given the following information. A function with the prototype *void decode1(int *xp, int *yp, int *zp)* is compiled into assembly code. The body of the code is as follows:

```
# xp at %ebp+8, yp at %ebp+12, zp at %ebp+16

movl 8(%ebp),%edi       x
movl 12(%ebp), %edx     y
movl 16(%ebp), %ecx     z
movl (%edx), %ebx    ebx = *y
movl (%ecx), %esi    esi = *z
movl (%edi), %eax    eax = *x
movl %eax, (%edx)    *y = *x
```

```
void de-code1 (int *xp, int *yp, int *zp){
    int a = *yp;
    *yp = *xp;
    *xp = *zp;
    *zp = a;
}
```

```
movl %ebx, (%ecx)  *z = *y
movl %esi, (%edi)  *x = *z
```

Write C code for decode1 that will have an effect equivalent to the assembly code above.

3. A function with prototype *int decode2(int x, int y, int z)* is compiled into the following IA32 code -

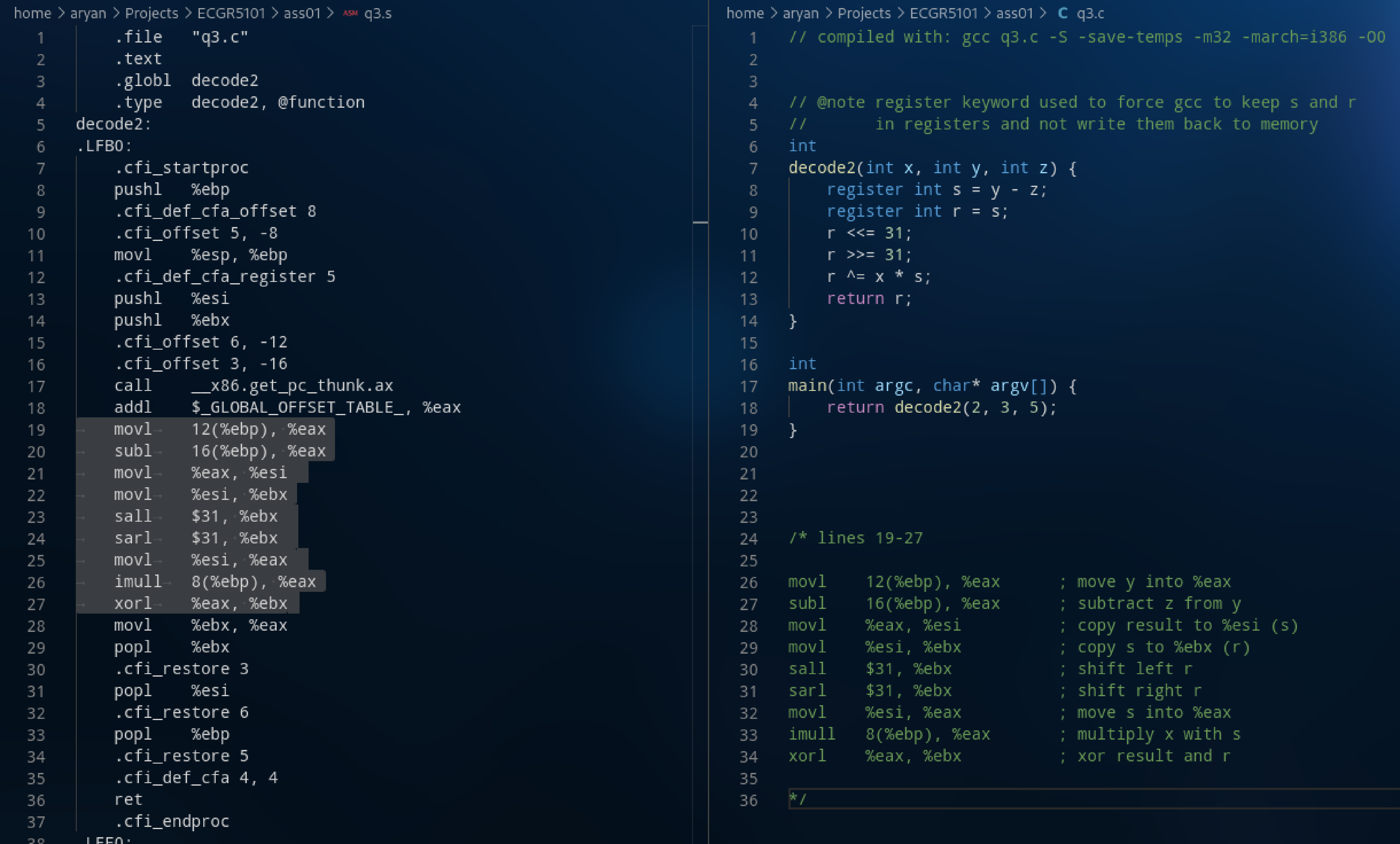
```
#x at %ebp+8, y at %ebp+12, z at %ebp+16

                          y
movl 12(%ebp), %edx
                          z
subl 16(%ebp), %edx  y-z
                        y
movl %edx, %eax
sall $31, %eax  y<<31
sarl $31, %eax  y>>31
             x            s
imull 8(%ebp), %edx  ~~s~~*x
xorl %edx, %eax  y* mul
```

Write C code for decode2 that will have an equivalent effect to the assembly code. Check your answer by compiling your code with gcc (flags -m32 and -march=i386) and examining the assembly.

```
int decode2(int x, int y, int z){
    int s = y-z;
    int r = s;
    r <<= 31;
    r >>= 31;
    r ^= x*s;
    return r;

}
```



home > aryan > Projects > ECGR5101 > ass01 > q3.s

```
 1      .file   "q3.c"
 2      .text
 3      .globl  decode2
 4      .type   decode2, @function
 5  decode2:
 6  .LFB0:
 7      .cfi_startproc
 8      pushl   %ebp
 9      .cfi_def_cfa_offset 8
10      .cfi_offset 5, -8
11      movl    %esp, %ebp
12      .cfi_def_cfa_register 5
13      pushl   %esi
14      pushl   %ebx
15      .cfi_offset 6, -12
16      .cfi_offset 3, -16
17      call    __x86.get_pc_thunk.ax
18      addl    $_GLOBAL_OFFSET_TABLE_, %eax
19      movl    12(%ebp), %eax
20      subl    16(%ebp), %eax
21      movl    %eax, %esi
22      movl    %esi, %ebx
23      sall    $31, %ebx
24      sarl    $31, %ebx
25      movl    %esi, %eax
26      imull   8(%ebp), %eax
27      xorl    %eax, %ebx
28      movl    %ebx, %eax
29      popl    %ebx
30      .cfi_restore 3
31      popl    %esi
32      .cfi_restore 6
33      popl    %ebp
34      .cfi_restore 5
35      .cfi_def_cfa 4, 4
36      ret
37      .cfi_endproc
38  .LFE0:
```

home > aryan > Projects > ECGR5101 > ass01 > q3.c

```c
// compiled with: gcc q3.c -S -save-temps -m32 -march=i386 -O0


// @note register keyword used to force gcc to keep s and r
//       in registers and not write them back to memory
int
decode2(int x, int y, int z) {
    register int s = y - z;
    register int r = s;
    r <<= 31;
    r >>= 31;
    r ^= x * s;
    return r;
}

int
main(int argc, char* argv[]) {
    return decode2(2, 3, 5);
}




/* lines 19-27

movl    12(%ebp), %eax      ; move y into %eax
subl    16(%ebp), %eax      ; subtract z from y
movl    %eax, %esi          ; copy result to %esi (s)
movl    %esi, %ebx          ; copy s to %ebx (r)
sall    $31, %ebx           ; shift left r
sarl    $31, %ebx           ; shift right r
movl    %esi, %eax          ; move s into %eax
imull   8(%ebp), %eax       ; multiply x with s
xorl    %eax, %ebx          ; xor result and r

*/
```