CSC 2400 – Translating X86 to C

Exercise 1 [Arithmetic Operations]

Consider the following assembly code for a function F1 with two integer arguments:

F1:

```assembly
push EBP
mov EBP, ESP        # setup stack
mov EDX, DWORD PTR [ebp+12]  # EDX = y
mov EAX, DWORD PTR [ebp+8]   # EAX = x
add EAX, EAX            # EAX = _____
add EAX, EDX             # EAX = _____
mov DWORD PTR [ebp-4], EAX # ignore for now
mov ESP, EBP             # cleanup stack
pop EBP
ret
```

To the right of each instruction, show the contents of the register whose value changes as a result of executing that instruction, then fill in the blanks in the corresponding C source code below.

Note: you may only use the symbolic variables \( x, y \) in your expressions below --- do not use register names.

```c
int F1(int x, int y)
{
    int result;

    result = ________________;

    return result;
}
```
Exercise 2 [Arithmetic Operations]

Consider the following assembly code for a function F2 with three integer arguments:

```
F1:
push EBP
mov EBP, ESP         # setup stack
mov ECX, DWORD PTR [ebp+16]  # ECX = z
mov EDX, DWORD PTR [ebp+12]  # EDX = y
mov EAX, DWORD PTR [ebp+8]   # EAX = x
add EAX, EDX           # EAX = _________
sal EAX, 2              # EAX = _________
sub EAX, ECX            # EAX = _________
mov DWORD PTR [ebp-4], EAX   # ignore for now
mov ESP, EBP            # cleanup stack
pop EBP
ret
```

To the right of each instruction, show the contents of the register whose value changes as a result of executing that instruction, then fill in the blanks in the corresponding C source code below.

```
int F2(int x, int y, int z)
{
    int result;
    result = ____________________;
    return result;
}
```
Exercise 3 [Conditionals]

Consider the following assembly code for a function F3 with two integer arguments:

F3:
```
push EBP            # setup stack
mov EBP, ESP
mov EDX, DWORD PTR [ebp+12]    # EDX = y
mov EAX, DWORD PTR [ebp+8]     # EAX = x
cmp EAX, EDX              # if ____________
jg .L1                     # goto .L1
mov EAX, EDX              # EAX = ____________
```

.L1:
```
mov DWORD PTR [EBP-4], EAX  # ignore for now
mov ESP, EBP              # cleanup stack
pop EBP
ret
```

To the right of each instruction, show the contents of the register whose value changes as a result of executing that instruction, then fill in the corresponding C source code below.

```c
int F3(int x, int y)
{
```
**Exercise 4 [Loops]**

Consider the following assembly code for a function `F4` containing a loop:

```
F4:
push EBP             # setup stack
mov EBP, ESP

mov EDX, DWORD PTR [ebp+12]       # EDX = y
mov ECX, DWORD PTR [ebp+8]        # ECX = x
xor EAX, EAX                   # EAX = _______
jmp .L2
.L1:
dec ECX                      # ___________
inc EDX                      # ___________
inc EAX                      # ___________
.L2:
cmp ECX, EDX                  # if _________
jg .L1                       # goto .L1
SAL EAX, 3                    # EAX = _______
mov DWORD PTR [EBP-4], EAX    # ignore for now

mov ESP, EBP                 # cleanup stack
pop EBP
ret
```

Complete the C code below corresponding to the assembly code above. You may only use the symbolic variables `x`, `y`, and `result` in your expressions below. *Do not use register names.*

```c
int F4(int x, int y)
{
    int result;

    return result;
}
```