Assembly and System Calls

Exercise 1. Using the "write" system call, write in assembly a print function that takes as input the address of a string and its length, and prints the string on the standard output. Use it to write a "Hello, World!" program.

Exercise 2. Write a function mystrlen that takes a zero-terminated string address as input and returns its length. You will need the "movzbl" instruction, which moves a single byte to its destination and zero-pads it.

Exercise 3. Write a function printz that takes a zero-terminated string address as input and prints this string on stdout.

Replace mystrlen by a call to the corresponding C function instead.

Exercise 4. Write *in OCaml* a function 'compile_aux' of type propf $\rightarrow$ (string $*$ int) list $\rightarrow$ unit that takes as input a propositional formula and an environment telling for each variable name its offset from %ebp, and prints on the standard output assembly code that evaluates the formula.

Exercise 5. Write *in OCaml* a function 'compile' that takes as input a propositional formula and outputs on the standard output a full assembly program that takes a valuation as command-line arguments and evaluates the formula. The following is an example of the output I have with this program:

```
.data
say:
        .string "Variable values:\n"
        .align  4
var:
        .string "  %s: %d\n"
        .align  4
res:
        .string "Result: %d\n"
        .align  4
x1:
        .asciz  "x1"
        .align  4
x2:
        .asciz  "x2"
        .align  4
x3:
        .asciz  "x3"
        .align  4

.text
        .global main
main:
```

```
.type   main, @function
pushl   %ebp
movl    %esp, %ebp
subl    $24, %esp       # make room for 3 pvars + 3 local vars
movl    12(%ebp), %eax  # load argv
movl    4(%eax), %eax   # load *argv[1]
movl    %eax, (%esp)
call    atoi
movl    %eax, -12(%ebp) # save x1
movl    12(%ebp), %eax  # load argv
movl    8(%eax), %eax   # load *argv[2]
movl    %eax, (%esp)
call    atoi
movl    %eax, -8(%ebp)  # save x2
movl    12(%ebp), %eax  # load argv
movl    12(%eax), %eax  # load *argv[3]
movl    %eax, (%esp)
call    atoi
movl    %eax, -4(%ebp)  # save x3
movl    $say,(%esp)
call    printf
movl    -12(%ebp), %eax
movl    %eax, 8(%esp)
movl    $x1, 4(%esp)
movl    $var, (%esp)
call    printf          # print x1
movl    -8(%ebp), %eax
movl    %eax, 8(%esp)
movl    $x2, 4(%esp)
movl    $var, (%esp)
call    printf          # print x2
movl    -4(%ebp), %eax
movl    %eax, 8(%esp)
movl    $x3, 4(%esp)
movl    $var, (%esp)
call    printf          # print x3
movl    -12(%ebp), %eax
pushl   %eax
movl    -8(%ebp), %eax
popl    %ebx
orl     %ebx, %eax
pushl   %eax
movl    -12(%ebp), %eax
pushl   %eax
movl    -4(%ebp), %eax
popl    %ebx
orl     %ebx, %eax
popl    %ebx
andl    %ebx, %eax
```

```
pushl   %eax
movl    -8(%ebp), %eax
pushl   %eax
movl    -4(%ebp), %eax
popl    %ebx
andl    %ebx, %eax
popl    %ebx
orl     %ebx, %eax
movl    %eax, 4(%esp)
movl    $res, (%esp)
call    printf
movl    4(%esp), %eax
call    exit
```