CS 241: Computer Organization and Assembly Language
Practice Final Exam

Do not open until instructed to do so.

Name: ____________________________

Sooner or later the world breaks everyone, and afterward many are strong in the broken places.
~Ernest Hemingway, *A Farewell to Arms*

Every problem is marked with a ► . When you see this symbol, it means that's a question which you can — and should — answer.

For grader use:

Score: ____

Syscalls

`0`	`sys_read`
`1`	`sys_write`
`60`	`sys_exit`

Arguments in: `rdi, rsi, rdx, r10, r8, r9` in that order

Return value in: `rax`

Clobbers: `rcx, r11`

Common syscalls

	1	Output	Addr.	Length
`write`	`rax`	`rdi`	`rsi`	`rdx`

	0	Input	Addr	Length
`read`	`rax`	`rdi`	`rsi`	`rdx`

	60	Exit code
`exit`	`rax`	`rdi`

C-style functions

```
func:
  push rbp
  mov rbp, rsp

  ...

  pop rbp
  ret
```

Arguments in: `rdi, rsi, rdx, rcx, r8, r9` in that order

Return value in: `rax`

Callee-saved regs.: `rbx, rbp, r12-r15`

Clobbers: `rax, r10, r11`, argument registers

`rsp` must be a multiple of 16, plus 8, before any `call`. `rsp` is a multiple of 16 on function entry.

Memory operands

`size [displacement + base + m * offset]` (with *size*, *displacement*, *base*, *offset* in italics)

`size` byte, word, dword, etc.

`displacement` Constant address of array

`base` Array base register

`m` 1, 2, 4, or 8

`offset` Array offset register

Instructions

`mov rm, rmi`	Move
`xchng rm, rm`	Swap
`lea r, m`	Load Effective Address
`xor r, r`	Set *r* to `0`
`add rm, rmi`	Addition
`sub rm, rmi`	Subtraction
`mul rmi`	Unsigned multiply (by/into `rax`)
`div rmi`	Unsigned divide (into `rax`)
`imul rmi`	Signed multiply
`idiv rmi`	Signed divide
`cmp rm, rmi`	Compare, update flags
`text rm, rmi`	Test, update flags
`jmp target`	Jump to target
`j`*`CC`* `target`	Jump if condition *CC*
`loop target`	Decrement `rcx`, jump if not `0`
`call func`	Push `rip`, jump to `func`
`ret`	Pop `rip` and jump to
`push rm`	Push onto stack
`pop rm`	Pop from stack

`r`: register, `m`: memory operand, `i`: immediate

Condition codes

CC	Meaning
`a`	Unsigned $>$
`ae`	Unsigned $\geq$
`b`	Unsigned $<$
`be`	Unsigned $\leq$
`g`	Signed $>$
`ge`	Signed $\geq$
`l`	Signed $<$
`le`	Signed $\leq$
`e`	$=$
`ne`	$\neq$
`s,c,z,...`	If flag is set

---

5 points each

► Perform the following binary addition:
`10110100 + 00111111`
Show your work (all carries).

► Suppose we want to swap the (byte) values in the registers `al` and `ah`. Write assembly code to do the swap.

► Perform the addition `01110100 + 10111111`, show your work, write the final sum, as well as the state of the flags after the addition is complete.

`CF =`

`OF =`

`SF =`

`ZF =`

▶ Perform the comparison `cmp 0b01110100, 0b10111111` and show the state of the flags after the comparison. (You can't actually do an immediate-immediate comparison, but just pretend.)

```
CF =


OF =


SF =


ZF =
```

▶ Write assembly equivalent to the following C code:

```
int rax, rdi, rbx;

if(rax > 0)
  if(rdi < 10)
    rbx = 0;
```

Suppose we have the following structure definition:

```
struct S {
  int a;
  long b;
  char c;
  char* d;
};
```

▶ What is the size of this structure in bytes?

▶ What are the offsets, in bytes, of each of the structure members from the beginning of the structure?

```
S::a


S::b


S::c


S::d
```

▶ Write assembly code using string instructions to copy a 100 byte array from the address in `rax` to the address in `rbx`.

Coding problems

You should create a directory on the server called `~/cs241/final/` for your answers to these problems.

The first two problems will replace the equivalent section from the midterm, if you do better here than there. If you are happy with your grade on the midterm, *you do not need to do these problems.*

► Complete the following `syscall`-style function so that it will print out a triangle made of * characters. E.g., if the function's parameter in `rsi` is 5, it should print out

```
*
**
***
****
*****
```

```
section .data

newline:    db    10
star:       db    '*'

section .text

print_stars:
  ; Your code here...
```

► Complete the following function so that it returns 1 if the qword array pointed to by `rdi` (array length in `rsi`) contains any duplicates, or `0` if it does not.

```
has_duplicates:
  ; Your code here
```

These problems are new to the final; you must work them to pass.

25 points each

► Complete the definition of a C-style function

```
void capitalize(char* s);
```

which converts all lower-case characters (those in ASCII range 97-122) to upper case (65-89) in the (nul-terminated) string s

```
capitalize:
  ; Your code here
```

► Complete the definition of a C-style function

```
long* find(long* array, unsigned long length, long value);
```

which takes a pointer to a signed qword array, a (qword) length, and a signed qword value, and returns either a pointer to the array element containing the value, or the null pointer if the value does not exist in the array.