```
.file "lab5.s"
.globl main
    .type    main, @function

.text
main:
    pushq %rbp        #stack housekeeping
    movq %rsp, %rbp

Label1:
#as you go through this program note the changes to %rip
    movq    $0x8877665544332211, %rax   # the value of %rax is: 0x8877665544332211, %rip: 0x401193
    movb    $-1, %al                    # the value of %rax is: 0x88776655443322ff, %rip: 0x401195
    movw    $-1, %ax                    # the value of %rax is: 0x887766554433ffff, %rip: 0x401199
    movl    $-1, %eax                   # the value of %rax is: 0x00000000ffffffff, %rip: 0x40119e
    movq    $-1, %rax                   # the value of %rax is: 0xffffffffffffffff, %rip: 0x4011a5

    movl    $-1, %eax                   # the value of %rax is: 0x00000000ffffffff, %rip: 0x4011aa
    cltq                                # the value of %rax is: 0xffffffffffffffff, %rip: 0x4011ac

    movl    $0x7fffffff, %eax           # the value of %rax is: 0x000000007fffffff, %rip: 0x4011b1
    cltq                                # the value of %rax is: 0x000000007fffffff, %rip: 0x4011b3
    movl    $0x8fffffff, %eax           # the value of %rax is: 0x000000008fffffff, %rip: 0x4011b8
    cltq                                # the value of %rax is: 0xffffffff8fffffff, %rip: 0x4011ba
                                        # what do you think the cltq instruction does? When cltq executed, the value in %eax were extended to %rax.

    movq    $0x8877665544332211, %rax   # the value of %rax is: 0x8877665544332211, %rip:0x4011c4

    # the value of %rdx *before* movb $0xAA, %dl executes is: 0x00007fffffffe3c8
    movb    $0xAA, %dl          # the value of %rdx is: 0x00007fffffffe3aa, %rip: 0x4011c6
    movb    %dl, %al            # the value of %rax is: 0x88776655443322aa, %rip: 0x4011c8
    movsbw    %dl, %ax          # the value of %rax is: 0x887766554433ffaa, %rip: 0x4011cc
    movzbw    %dl, %ax          # the value of %rax is: 0x8877665544330aa, %rip: 0x4011d0

    movq    $0x8877665544332211, %rax    # the value of %rax is: 0x8877665544332211, %rip: 0x4011da
    movb    %dl, %al            # the value of %rax is: 0x88776655443322aa, %rip: 0x4011dc
    movsbl    %dl, %eax         # the value of %rax is: 0x00000000ffffffaa, %rip: 0x4011df
    movzbl    %dl, %eax         # the value of %rax is: 0x000000000000000aa, %rip: 0x4011e2

    movq    $0x8877665544332211, %rax    # the value of %rax is: 0x8877665544332211, %rip: 0x4011ec
    movb    %dl, %al            # the value of %rax is: 0x88776655443322aa, %rip: 0x4011ee
    movsbq  %dl, %rax           # the value of %rax is: 0xffffffffffffffaa, %rip: 0x4011f2
    movzbq  %dl, %rax           # the value of %rax is: 0x000000000000000aa, %rip: 0x4011f6

    movq    $0x8877665544332211, %rax    # the value of %rax is: 0x8877665544332211, %rip: 0x401200
    # the value of %rdx *before* movb $0x0000000000000055, %dl executes is: 0xaa
    movb    $0x55, %dl          # the value of %rdx is: 0x8877665544332211, %rip: 0x401202
    movb    %dl, %al            # the value of %rax is: 0x8877665544332255, %rip: 0x401204
    movsbw  %dl, %ax            # the value of %rax is: 0x8877665544330055, %rip: 0x401208
    movzbw  %dl, %ax            # the value of %rax is: 0x8877665544330055, %rip: 0x40120c

    movq    $0x8877665544332211, %rax # the value of %rax is: 0x8877665544332211, %rip: 0x401216
    movb    %dl, %al            # the value of %rax is: 0x8877665544332255, %rip: 0x401218
    movsbl  %dl, %eax           # the value of %rax is: 0x0000000000000055, %rip: 0x40121b
    movzbl  %dl, %eax           # the value of %rax is: 0x0000000000000055, %rip: 0x40121e

    movq    $0x8877665544332211, %rax # the value of %rax is: 0x8877665544332211, %rip: 0x401228
    movb    %dl, %al            # the value of %rax is: 0x8877665544332255, %rip: 0x40122a
    movsbq  %dl, %rax           # the value of %rax is: 0x0000000000000055, %rip: 0x40122e
    movzbq  %dl, %rax           # the value of %rax is: 0x0000000000000055, %rip: 0x401232

#    movq    $0x8877665544332211, %rax
#    pushb   %al
#    movq    $0, %rax
#    popb    %al

    movq    $0x8877665544332211, %rax    # the value of %rax is: 0x8877665544332211, the value of %rsp is: 0x00007fffffffe2d0
    pushw   %ax                          # the value of %rsp is: 0x00007fffffffe2ce
                                         # the difference between the two values of %rsp is: 2
    movq    $0, %rax                     # the value of %rax is: 0x0000000000000000
    popw    %ax                          # the value of %rax is: 0x0000000000002211, How did the value of %rsp change? 0x00007fffffffe2d0, added by 2

    movq    $0x8877665544332211, %rax    # the value of %rax is: 0x8877665544332211, the value of %rsp is: 0x00007fffffffe2d0
    pushw   %ax                          # the value of %rsp is: 0x00007fffffffe2ce
                                         # the difference between the two values of %rsp is: 2
    movq    $-1, %rax                    # the value of %rax is: 0xffffffffffffffff
    popw    %ax                          # the value of %rax is: 0xffffffffffff2211, How did the value of %rsp change? 0x00007fffffffe2d0, returned to original value


#    movq    $0x8877665544332211, %rax
#    pushl   %eax
#    movq    $0, %rax
#    popl    %eax

    movq    $0x8877665544332211, %rax    # the value of %rax is: 0x8877665544332211, the value of %rsp is: 0x7fffffffe2d0
    pushq   %rax                         # the value of %rsp is: 0x00007fffffffe2c8
                                         # the difference between the two values of %rsp is: 8
    movq    $0, %rax                     # the value of %rax is: 0x0
    popq    %rax                         # the value of %rax is: 0x8877665544332211, How did the value of %rsp change? Changed back to 0x00007fffffffe2d0

                                         # what eflags are set? 0x246    [ PF ZF IF ]

    movq    $0x500, %rax                 # the value of %rax is: 0x0000000000000500
    movq    $0x123, %rcx                 # the value of %rcx is: 0x0000000000000123
    # 0x123 - 0x500
    subq    %rax, %rcx                   # the value of %rax is: 0x0000000000000500
            # the value of %rcx is: 0xfffffffffffffc23

          # what eflags are set? 0x283    [ CF SF IF ]

    movq    $0x500, %rax                 # the value of %rax is: 0x0000000000000500
    movq    $0x123, %rcx                 # the value of %rcx is: 0x0000000000000123
    # 0x500 - 0x123
    subq    %rcx, %rax                   # the value of %rax is: 0x00000000000003dd
                  # what eflags are set? 0x216    [ PF AF IF ]

    movq    $0x500, %rax                 # the value of %rax is: 0x0000000000000500
    movq    $0x500, %rcx                 # the value of %rcx is: 0x0000000000000500
    # 0x500 - 0x500
    subq    %rcx, %rax                   # the value of %rax is: 0x0000000000000000
                # what eflags are set? 0x246    [ PF ZF IF ]

    movb    $0xff, %al      # the value of %rax is: 0x00000000000000ff
    # 0xff +=1 (1 byte)
    incb    %al             # the value of %rax is: 0x0000000000000000, what eflags are set? 0x256    [ PF AF ZF IF ]

    movb    $0xff, %al      # the value of %rax is: 0x00000000000000ff
    # 0xff +=1 (4 bytes)
    incl    %eax                # the value of %rax is: 0x0000000000000100, what eflags are set? 0x216    [ PF AF IF ]

    movq    $-1, %rax           # the value of %rax is: 0xffffffffffffffff
    # 0xff +=1 (8 bytes)
    incq    %rax                # the value of %rax is: 0x0000000000000000, what eflags are set? 0x256    [ PF AF ZF IF ]

    movq    $0x8877665544332211, %rax    # the value of %rax is: 0x8877665544332211
    movq    $0x8877665544332211, %rcx    # the value of %rax is: 0x8877665544332211, what eflags are set? 0x256    [ PF AF ZF IF ]
    addq    %rcx, %rax                   # the value of %rax is: 0x10eeccaa88664422, what eflags are set? 0xa07    [ CF PF IF OF ]

    movq    $0x8877665544332211, %rax    # the value of %rax is: 0x8877665544332211
    andq    $0x1, %rax                   # the value of %rax is: 0x0000000000000001

    movq    $0x8877665544332211, %rax    # the value of %rax is: 0x8877665544332211, explain why the values for AND/OR/XOR are
    andq    %rax, %rax                   # the value of %rax is: 0x8877665544332211, what they are
    orq     %rax, %rax                     # the value of %rax is: 0x8877665544332211
    xorq    %rax, %rax                   # the value of %rax is: 0x0000000000000000

    movq    $0x8877665544332211, %rax    # the value of %rax is: 0x8877665544332211
    andw    $0x3300, %ax                 # the value of %rax is: 0x8877665544332200, explain the value in the 8 byte register vs
                                         # the value in the 2 byte register

    salq    $4, %rax                        # the value of %rax is: 0x8776655443322000, Why? We moved 4 bits to left and the missing bits were filled by 0 at last.

    movq    $0xff0000001f000000, %rax   # the value of %rax is: 0xff0000001f000000, what do these 6 values look like in binary???      1111 1111 0000 0000 0000 0000 0000 0000 0001 1111 0000 0000 0000 0000 0000 0000
    sall    $1, %eax                    # the value of %rax is: 0x000000003e000000  do these shift instructions do what you expected?  0000 0000 0000 0000 0000 0000 0000 0000 0001 1111 0000 0000 0000 0000 0000 0000
    sall    $1, %eax                    # the value of %rax is: 0x000000007c000000  The binary value did the left shift as expected.   0000 0000 0000 0000 0000 0000 0000 0000 0011 1110 0000 0000 0000 0000 0000 0000
    sall    $1, %eax                    # the value of %rax is: 0x00000000f8000000  Left shift starts from %eax.                       0000 0000 0000 0000 0000 0000 0000 0000 0111 1100 0000 0000 0000 0000 0000 0000
    sall    $1, %eax                    # the value of %rax is: 0x00000000f0000000                                                     0000 0000 0000 0000 0000 0000 0000 0000 1111 1000 0000 0000 0000 0000 0000 0000
    sall    $1, %eax                    # the value of %rax is: 0x00000000e0000000                                                     0000 0000 0000 0000 0000 0000 0000 0000 1111 0000 0000 0000 0000 0000 0000 0000

    movq    $0xff000000ff000000, %rax   # the value of %rax is: 0xff000000ff000000, what do these 6 values look like in binary???      1111 1111 0000 0000 0000 0000 0000 0000 1111 1111 0000 0000 0000 0000 0000 0000
    salq    $1, %rax                    # the value of %rax is: 0xfe000001fe000000  do these shift instructions do what you expected?  1111 1110 0000 0000 0000 0000 0000 0001 1111 1110 0000 0000 0000 0000 0000 0000
    salq    $1, %rax                    # the value of %rax is: 0xfc000003fc000000  The binary value did the left shift as expected.   1111 1100 0000 0000 0000 0000 0000 0011 1111 1100 0000 0000 0000 0000 0000 0000
    salq    $1, %rax                    # the value of %rax is: 0xf8000007f8000000  Left shift starts from %rax.                       1111 1000 0000 0000 0000 0000 0000 0111 1111 1000 0000 0000 0000 0000 0000 0000
    salq    $1, %rax                    # the value of %rax is: 0xf000000ff0000000                                                     1111 0000 0000 0000 0000 0000 0000 1111 1111 0000 0000 0000 0000 0000 0000 0000
    salq    $1, %rax                    # the value of %rax is: 0xe000001fe0000000                                                     1110 0000 0000 0000 0000 0000 0001 1111 1110 0000 0000 0000 0000 0000 0000 0000

    movq    $0xff0000000000ff, %rax   # the value of %rax is: 0xff0000000000ff, what do these 6 values look like in binary???      1111 1111 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 1111 1111
```

```
158     sarq    $1, %rax                    # the value of %rax is: 0xff8000000000007f  do these shift instructions do what you expected?  1111 1111 1000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0111 1111
159     sarq    $1, %rax                    # the value of %rax is: 0xffc000000000003f  The binary value did the left shift as expected.   1111 1111 1100 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0011 1111
160     sarq    $1, %rax                    # the value of %rax is: 0xffe000000000001f  Arithmetic right shift starts from %rax, missing   1111 1111 1110 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0001 1111
161     sarq    $1, %rax                    # the value of %rax is: 0xfff000000000000f  bits were filled by sign at left.                 1111 1111 1111 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 1111
162     sarq    $1, %rax                    # the value of %rax is: 0xfff8000000000007                                                  1111 1111 1111 1000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0111
163
164     movq    $0xff000000000000ff, %rax   # the value of %rax is: 0xff000000000000ff, what do these 6 values look like in binary???      1111 1111 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 1111 1111
165     shrq    $1, %rax                    # the value of %rax is: 0x7f8000000000007f  do these shift instructions do what you expected?  0111 1111 1000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0111 1111
166     shrq    $1, %rax                    # the value of %rax is: 0x3fc000000000003f  The binary value did the left shift as expected.   0011 1111 1100 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0011 1111
167     shrq    $1, %rax                    # the value of %rax is: 0x1fe000000000001f  Logic right shift starts from %rax, missing bits   0001 1111 1110 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0001 1111
168     shrq    $1, %rax                    # the value of %rax is: 0x0ff000000000000f  were filled by 0 at left.                         0000 1111 1111 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 1111
169     shrq    $1, %rax                    # the value of %rax is: 0x07f8000000000007                                                  0000 0111 1111 1000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0111
170
171     movq    $0xff000000000000ff, %rax   # the value of %rax is: 0xff000000000000ff, what do these 6 values look like in binary???      1111 1111 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 1111 1111
172     sarw    $1, %ax                     # the value of %rax is: 0xff0000000000007f  do these shift instructions do what you expected?  1111 1111 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0111 1111
173     sarw    $1, %ax                     # the value of %rax is: 0xff0000000000003f  The binary value did the left shift as expected.   1111 1111 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0011 1111
174     sarw    $1, %ax                     # the value of %rax is: 0xff0000000000001f  Arithmetic right shift starts from %ax, missing    1111 1111 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0001 1111
175     sarw    $1, %ax                     # the value of %rax is: 0xff0000000000000f  bits were filled by sign at left.                 1111 1111 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 1111
176     sarw    $1, %ax                     # the value of %rax is: 0xff0000000000007                                                   1111 1111 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0111
177
178     movq    $0xff000000000000ff, %rax   # the value of %rax is: 0xff000000000000ff, what do these 6 values look like in binary???      1111 1111 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 1111 1111
179     shrw    $1, %ax                     # the value of %rax is: 0xff0000000000007f, do these shift instructions do what you expected?  1111 1111 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0111 1111
180     shrw    $1, %ax                     # the value of %rax is: 0xff0000000000003f  The binary value did the left shift as expected.   1111 1111 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0011 1111
181     shrw    $1, %ax                     # the value of %rax is: 0xff0000000000001f  Logic right shift starts from %ax, missing bits   1111 1111 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0001 1111
182     shrw    $1, %ax                     # the value of %rax is: 0xff0000000000000f  were filled by 0 at left.                         1111 1111 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 1111
183     shrw    $1, %ax                     # the value of %rax is: 0xff0000000000007                                                   1111 1111 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0111
184
185
186     leave                       #post function stack cleanup
187     ret
188
189 .size main, .-main
```