CSE 2421 SP19 Lab 5

bits.s

Conventional bits.s (1 of 2)

```
.text
bits:
        pushq %rbp
        movq %rsp, %rbp

        xorq %rax, %rax         # zero return value

        subq %rsi, %rsi         # scratch pad to hold the 1 or 0
        movq $1, %rdx           # the traveling bit lives here
        xorq %rcx, %rcx         # zero the current count
```

Conventional bits.s (2 of 2)

```
loop_start:
        testq %rdx, %rdi        #probe the number with the traveling bit
        setne %sil              # sets no flags, so Z stays good
        cmove %rsi, %rcx        #if that was a zero, clear the current count
        addq %rsi, %rcx         # sum into current (rsi has either 0 or 1)

        cmp %rax, %rcx          # is current larger than max?
        cmovg %rcx, %rax        # overwrite if current > max

        shlq %rdx               # shift the traveling bit
        jne loop_start          #keep going until that bit falls off the left end

        leave
        ret
```

Alternative bits.s (from Sanja Kopitar) [setup]

```
bits:
        pushq %rbp          #setting up the stack
        movq %rsp, %rbp     #frame

        #rax is count register
        #rdi is the input binary number (we will overwrite it) “the number”
        #rdx is that number shifted one place “the copy”

        movq $0, %rax               #zero the total count
        movq %rdi, %rdx     # write the number to the copy

        testq %rdi, %rdi    #if the number has no set bits we are done here
        jz loop_end
```

Alternative bits.s [loop]

```
loop_start:
        incq %rax           #increment the count if there were bits
                            #we have at least one bit or we'd not be here

        shlq %rdx           #shift the copy by one bit
        andq %rdi, %rdx     #look for overlapping bits between the
                            #number and the shifted copy
        movq %rdx, %rdi     #the resulting overlap is the new
                            #number to test.

        jnz loop_start      #do it again if we had overlapping bits
```

Alternative bits.s [cleanup]

- loop_end:
- 	leave		#unroll stack frame
- 	ret		#return the count of consecutive bits in rax