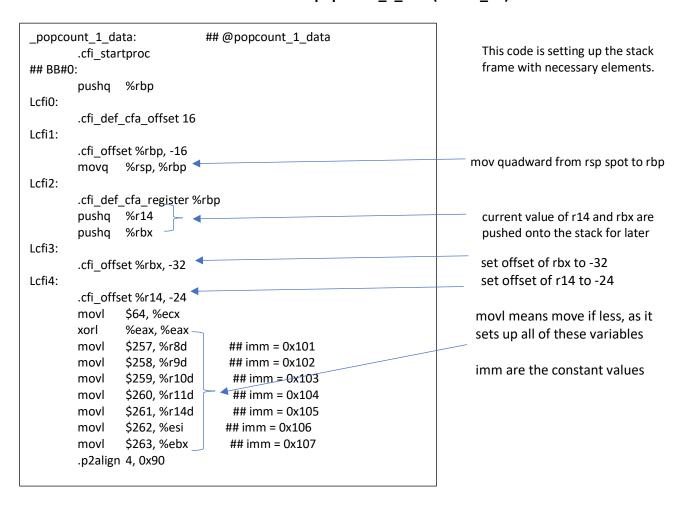
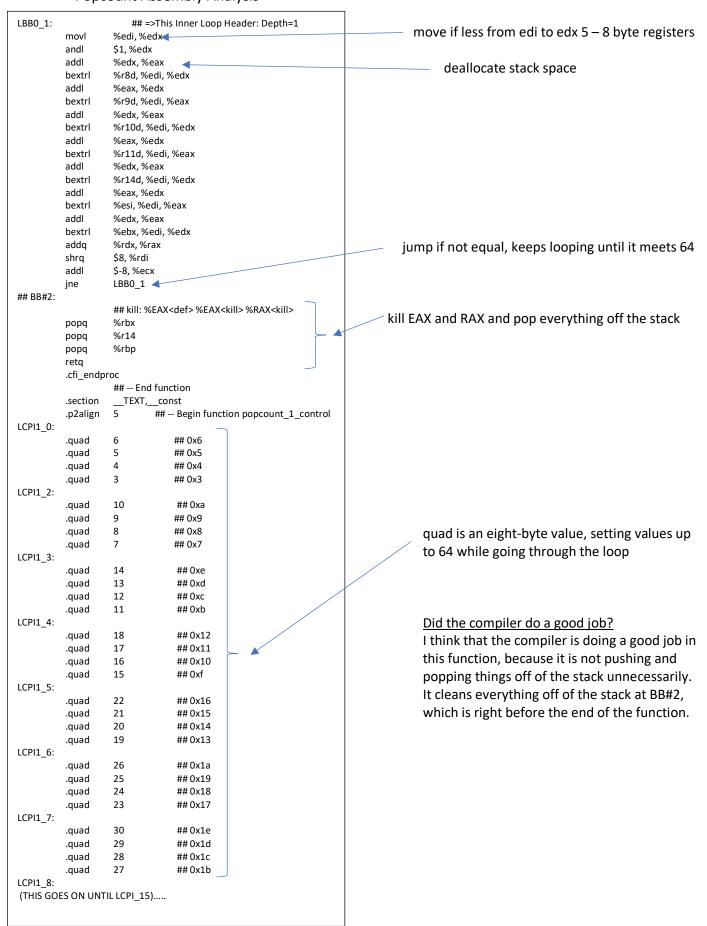
#### FUNCTION: int popcount\_1\_data(uint64\_t x)

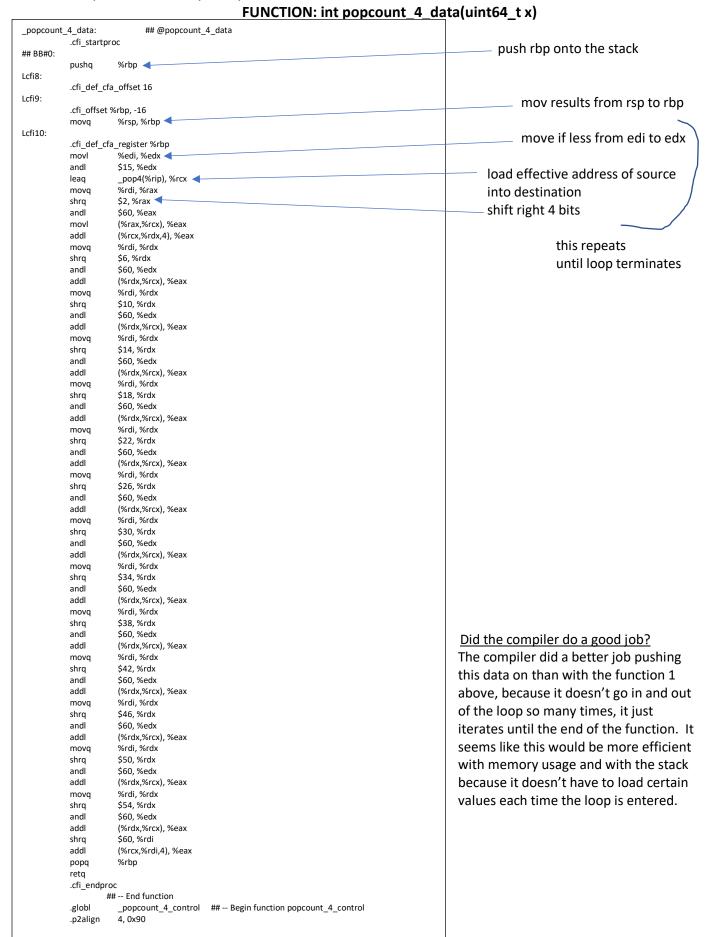


# Kayla Cresswell CS 6015 Popcount Assembly Analysis



## Kayla Cresswell CS 6015

#### **Popcount Assembly Analysis**



### Kayla Cresswell CS 6015

#### Popcount Assembly Analysis

### FUNCTION: int popcount\_1\_control(uint64\_t x)

```
_popcount_1_control:
                            ## @popcount_1_control
         .cfi_startproc
## BB#0:
                  %rbp 🚤
         pushq
Lcfi5:
         .cfi_def_cfa_offset 16
Lcfi6:
         .cfi_offset %rbp, -16
                  %rsp, %rbp
         mova
Lcfi7:
         .cfi def cfa register %rbp
                  %edi, %ecx 🔺
         movl
         andl
                  $1, %ecx
                                  ## imm = 0x101
         movl
                  $257, %eax
                  %eax, %edi, %r8d
         bextrl
         movl
                  $258, %eax
                                  ## imm = 0x102
                  %eax, %edi, %r9d
         bextrl
                  %rdi, %xmm0◀
         vmovq
                           %xmm0. %vmm3 <
         vpbroadcastq
                  LCPI1_0(%rip), %ymm3, %ymm0
         vpsrlvq
         vpshufd
                  $232, %ymm0, %ymm0
                                       ## ymm0 = ymm0[0,2,2,3,4,6,6,7] <
                                        ## ymm0 = ymm0[0,2,2,3]
                  $232, %ymm0, %ymm0
         vperma
                  LCPI1 1(%rip), %eax
         movl
         vmovd
                  %eax, %xmm1
                           %xmm1. %xmm2
         vpbroadcastd
         vpand
                  %xmm2, %xmm0, %xmm0
                  LCPI1_2(%rip), %ymm3, %ymm2
         vpsrlva
                  LCPI1_3(%rip), %ymm3, %ymm4
         vpsrlva
                  $232, %ymm4, %ymm4 ## ymm4 = ymm4[0,2,2,3,4,6,6,7]
         vpshufd
                  $232, %ymm4, %ymm4
                                        ## ymm4 = ymm4[0,2,2,3]
         vpermq
         vpshufd
                  $232, %ymm2, %ymm2
                                        ## ymm2 = ymm2[0,2,2,3,4,6,6,7]
                  $232, %ymm2, %ymm2 ## ymm2 = ymm2[0,2,2,3]
         vpermq
                           $1, %xmm2, %ymm4, %ymm4
         vinserti128
         vpbroadcastd
                           %xmm1, %ymm2
                  %ymm2, %ymm4, %ymm1
         vpand
         vpsrlva
                  LCPI1 4(%rip), %ymm3, %ymm4
                  LCPI1_5(%rip), %ymm3, %ymm5
         vpsrlvq
         vpsrlvq
                  LCPI1_6(%rip), %ymm3, %ymm6
         vpsrlvq
                  LCPI1_7(%rip), %ymm3, %ymm7
                  $232, %ymm7, %ymm7
                                        ## ymm7 = ymm7[0,2,2,3,4,6,6,7]
         vpshufd
                  $232, %ymm7, %ymm7
         vpermq
                                        ## ymm7 = ymm7[0,2,2,3]
                                        ## ymm6 = ymm6[0,2,2,3,4,6,6,7]
                  $232, %ymm6, %ymm6
         vpshufd
         vpermq
                  $232, %ymm6, %ymm6
                                        ## ymm6 = ymm6[0,2,2,3]
         vinserti128
                           $1, %xmm6, %ymm7, %ymm6
                                        ## ymm5 = ymm5[0,2,2,3,4,6,6,7]
         vpshufd $232, %ymm5, %ymm5
                  $232, %ymm5, %ymm5
                                        ## ymm5 = ymm5[0,2,2,3]
         vpermq
                  $232, %ymm4, %ymm4
                                        ## ymm4 = ymm4[0,2,2,3,4,6,6,7]
         vpshufd
                                        ## ymm4 = ymm4[0,2,2,3]
                  $232, %ymm4, %ymm4
         vperma
         vinserti128
                           $1, %xmm4, %ymm5, %ymm4
                  %ymm2, %ymm4, %ymm4
         vpand
         vpand
                  %ymm2, %ymm6, %ymm5
                  LCPI1_8(%rip), %ymm3, %ymm6
         vpsrlvq
                  LCPI1_9(%rip), %ymm3, %ymm7
         vpsrlvq
         vpsrlvq
                  LCPI1_10(%rip), %ymm3, %ymm8
                  LCPI1_11(%rip), %ymm3, %ymm9
         vpsrlvq
         vpsrlvq
                  LCPI1_14(%rip), %ymm3, %ymm10
                  LCPI1_15(%rip), %ymm3, %ymm11
         vpsrlva
                  $232, %ymm11, %ymm11 ## ymm11 =
         vpshufd
ymm11[0,2,2,3,4,6,6,7]
                  $232, %ymm11, %ymm11 ## ymm11 = ymm11[0,2,2,3]
         vpermq
                  $232, %ymm10, %ymm10 ## ymm10 =
         vpshufd
ymm10[0,2,2,3,4,6,6,7]
         vpermq $232, %ymm10, %ymm10 ## ymm10 = ymm10[0,2,2,3]
         vinserti128
                           $1, %xmm10, %ymm11, %ymm10
         vpsrlvq LCPI1_12(%rip), %ymm3, %ymm11
                  LCPI1 13(%rip), %ymm3, %ymm3
         vpsrlva
                  $232, %ymm3, %ymm3 ## ymm3 = ymm3[0,2,2,3,4,6,6,7]
                  $232, %ymm3, %ymm3 ## ymm3 = ymm3[0,2,2,3]
         vperma
```

\$232, %ymm11, %ymm11 ## ymm11 =

vpshufd

move edi if it is less

push rbp on the stack

mov quadward rdi to xmm0

shuffle packed bytes (x & 0xf code) bit shift right 4

load integer and broadcast

set immediate constant

....repeat till loop terminates....

Did the compiler do a good job?

I think that the compiler did a good job with this function as well. It is setting all the values, then permutating and shuffling, then inserting and finally broadcasting. It does a better job than setting the data in 1 because the LCPI1\_... through 15 is doing the bit shifting all in line right after each other. This seems very efficient to memory and with data accesses from RAM or from disk.