-WOTAN\PierrC:\Development\GitHub\Julia\Learning\20\_asm julia .\analyze\_count\_char.jl

----------------------------------------------------------- CountChar

**function** CountChar(s**::**String, letter**::**Char)**::**Int

    res **=** 0

**for** c **in** s

**if** c **==** letter

            res **+=** 1

**end**

**end**

    res

**end**

--------- @code\_warntype CountChar

MethodInstance for CountChar(::String, ::Char)

from CountChar(s::**String**, letter::**Char**) @ Main C:\Development\GitHub\Julia\Learning\20\_asm\analyze\_count\_char.jl:6

Arguments

#self#::Core.Const(CountChar)

s::String

letter::Char

Locals

@\_4**::Union{Nothing, Tuple{Char, Int64}}**

res::Int64

c::Char

@\_7::Int64

Body::Int64

1 ─ %1 = Main.Int::Core.Const(Int64)

│ (res = 0)

│ %3 = s::String

│ (@\_4 = Base.iterate(%3))

│ %5 = (@\_4 === nothing)::Bool

│ %6 = Base.not\_int(%5)::Bool

└── goto #6 if not %6

2 ┄ %8 = @\_4::Tuple{Char, Int64}

│ (c = Core.getfield(%8, 1))

│ %10 = Core.getfield(%8, 2)::Int64

│ %11 = (c == letter)::Bool

└── goto #4 if not %11

3 ─ (res = res + 1)

4 ┄ (@\_4 = Base.iterate(%3, %10))

│ %15 = (@\_4 === nothing)::Bool

│ %16 = Base.not\_int(%15)::Bool

└── goto #6 if not %16

5 ─ goto #2

6 ┄ (@\_7 = res)

│ %20 = (@\_7 isa %1)::Core.Const(true)

└── goto #8 if not %20

7 ─ goto #9

8 ─ Core.Const(:(Base.convert(%1, @\_7)))

└── Core.Const(:(@\_7 = Core.typeassert(%23, %1)))

9 ┄ return @\_7

--------- @code\_lowered CountChar

CodeInfo(

1 ─ %1 = Main.Int

│ res = 0

│ %3 = s

│ @\_4 = Base.iterate(%3)

│ %5 = @\_4 === nothing

│ %6 = Base.not\_int(%5)

└── goto #6 if not %6

2 ┄ %8 = @\_4

│ c = Core.getfield(%8, 1)

│ %10 = Core.getfield(%8, 2)

│ %11 = c == letter

└── goto #4 if not %11

3 ─ res = res + 1

4 ┄ @\_4 = Base.iterate(%3, %10)

│ %15 = @\_4 === nothing

│ %16 = Base.not\_int(%15)

└── goto #6 if not %16

5 ─ goto #2

6 ┄ @\_7 = res

│ %20 = @\_7 isa %1

└── goto #8 if not %20

7 ─ goto #9

8 ─ %23 = Base.convert(%1, @\_7)

└── @\_7 = Core.typeassert(%23, %1)

9 ┄ return @\_7

)

--------- @code\_typed CountChar

CodeInfo(

1 ── %1 = Core.sizeof(s)::Int64

│ %2 = Base.sle\_int(0, %1)::Bool

│ %3 = Base.bitcast(UInt64, %1)::UInt64

│ %4 = Base.ult\_int(0x0000000000000000, %3)::Bool

│ %5 = Base.and\_int(%2, %4)::Bool

└─── goto #13 if not %5

2 ── goto #9 if not false

3 ── %8 = Core.sizeof(s)::Int64

│ %9 = Base.sle\_int(1, %8)::Bool

└─── goto #5

4 ── nothing::Nothing

5 ┄─ goto #7 if not %9

6 ── goto #8

7 ── %14 = invoke Base.BoundsError(s::Any, 1::Int64)::BoundsError

│ Base.throw(%14)::Union{}

└─── unreachable

8 ── nothing::Nothing

9 ┄─ %18 = $(Expr(:gc\_preserve\_begin, Core.Argument(2)))

│ %19 = $(Expr(:foreigncall, :(:jl\_string\_ptr), Ptr{UInt8}, svec(Any), 0, :(:ccall), Core.Argument(2)))::Ptr{UInt8}

│ %20 = Core.bitcast(Core.UInt, %19)::UInt64

│ %21 = Base.add\_ptr(%20, 0x0000000000000001)::UInt64

│ %22 = Core.bitcast(Ptr{UInt8}, %21)::Ptr{UInt8}

│ %23 = Core.bitcast(Core.UInt, %22)::UInt64

│ %24 = Base.sub\_ptr(%23, 0x0000000000000001)::UInt64

│ %25 = Core.bitcast(Ptr{UInt8}, %24)::Ptr{UInt8}

│ %26 = Base.pointerref(%25, 1, 1)::UInt8

│ $(Expr(:gc\_preserve\_end, :(%18)))

└─── goto #10

10 ─ %29 = Core.zext\_int(Core.UInt32, %26)::UInt32

│ %30 = Base.shl\_int(%29, 0x0000000000000018)::UInt32

│ %31 = Base.lshr\_int(%29, 0xffffffffffffffe8)::UInt32

│ %32 = Core.ifelse(true, %30, %31)::UInt32

│ %33 = Base.ule\_int(0x80, %26)::Bool

│ %34 = Base.ule\_int(%26, 0xf7)::Bool

│ %35 = Base.and\_int(%33, %34)::Bool

└─── goto #12 if not %35

11 ─ %37 = invoke Base.iterate\_continued(s::String, 1::Int64, %32::UInt32)::Tuple{Char, Int64}

└─── goto #14

12 ─ %39 = Base.bitcast(Char, %32)::Char

│ %40 = Core.tuple(%39, 2)::Tuple{Char, Int64}

└─── goto #14

13 ─ goto #14

14 ┄ %43 = φ (#11 => false, #12 => false, #13 => true)::Bool

│ %44 = φ (#11 => %37, #12 => %40, #13 => Base.nothing)::Union{Nothing, Tuple{Char, Int64}}

└─── goto #15

15 ─ %46 = Base.not\_int(%43)::Bool

└─── goto #34 if not %46

16 ┄ %48 = φ (#15 => %44, #33 => %108)::Tuple{Char, Int64}

│ %49 = φ (#15 => 0, #33 => %57)::Int64

│ %50 = Core.getfield(%48, 1)::Char

│ %51 = Core.getfield(%48, 2)::Int64

│ %52 = Base.bitcast(Base.UInt32, %50)::UInt32

│ %53 = Base.bitcast(Base.UInt32, letter)::UInt32

│ %54 = (%52 === %53)::Bool

└─── goto #18 if not %54

17 ─ %56 = Base.add\_int(%49, 1)::Int64

18 ┄ %57 = φ (#17 => %56, #16 => %49)::Int64

│ %58 = Base.bitcast(UInt64, %51)::UInt64

│ %59 = Base.sub\_int(%58, 0x0000000000000001)::UInt64

│ %60 = Core.sizeof(s)::Int64

│ %61 = Base.sle\_int(0, %60)::Bool

│ %62 = Base.bitcast(UInt64, %60)::UInt64

│ %63 = Base.ult\_int(%59, %62)::Bool

│ %64 = Base.and\_int(%61, %63)::Bool

└─── goto #31 if not %64

19 ─ goto #27 if not false

20 ─ %67 = Base.sle\_int(1, %51)::Bool

└─── goto #22 if not %67

21 ─ %69 = Core.sizeof(s)::Int64

│ %70 = Base.sle\_int(%51, %69)::Bool

└─── goto #23

22 ─ goto #23

23 ┄ %73 = φ (#21 => %70, #22 => false)::Bool

└─── goto #25 if not %73

24 ─ goto #26

25 ─ %76 = invoke Base.BoundsError(s::Any, %51::Int64)::BoundsError

│ Base.throw(%76)::Union{}

└─── unreachable

26 ─ nothing::Nothing

27 ┄ %80 = $(Expr(:gc\_preserve\_begin, Core.Argument(2)))

│ %81 = $(Expr(:foreigncall, :(:jl\_string\_ptr), Ptr{UInt8}, svec(Any), 0, :(:ccall), Core.Argument(2)))::Ptr{UInt8}

│ %82 = Core.bitcast(Core.UInt, %81)::UInt64

│ %83 = Base.bitcast(UInt64, %51)::UInt64

│ %84 = Base.add\_ptr(%82, %83)::UInt64

│ %85 = Core.bitcast(Ptr{UInt8}, %84)::Ptr{UInt8}

│ %86 = Core.bitcast(Core.UInt, %85)::UInt64

│ %87 = Base.sub\_ptr(%86, 0x0000000000000001)::UInt64

│ %88 = Core.bitcast(Ptr{UInt8}, %87)::Ptr{UInt8}

│ %89 = Base.pointerref(%88, 1, 1)::UInt8

│ $(Expr(:gc\_preserve\_end, :(%80)))

└─── goto #28

28 ─ %92 = Core.zext\_int(Core.UInt32, %89)::UInt32

│ %93 = Base.shl\_int(%92, 0x0000000000000018)::UInt32

│ %94 = Base.lshr\_int(%92, 0xffffffffffffffe8)::UInt32

│ %95 = Core.ifelse(true, %93, %94)::UInt32

│ %96 = Base.ule\_int(0x80, %89)::Bool

│ %97 = Base.ule\_int(%89, 0xf7)::Bool

│ %98 = Base.and\_int(%96, %97)::Bool

└─── goto #30 if not %98

29 ─ %100 = invoke Base.iterate\_continued(s::String, %51::Int64, %95::UInt32)::Tuple{Char, Int64}

└─── goto #32

30 ─ %102 = Base.bitcast(Char, %95)::Char

│ %103 = Base.add\_int(%51, 1)::Int64

│ %104 = Core.tuple(%102, %103)::Tuple{Char, Int64}

└─── goto #32

31 ─ goto #32

32 ┄ %107 = φ (#29 => false, #30 => false, #31 => true)::Bool

│ %108 = φ (#29 => %100, #30 => %104, #31 => Base.nothing)::Union{Nothing, Tuple{Char, Int64}}

│ %109 = Base.not\_int(%107)::Bool

└─── goto #34 if not %109

33 ─ goto #16

34 ┄ %112 = φ (#32 => %57, #15 => 0)::Int64

└─── return %112

) => Int64

--------- @code\_llvm CountChar

; @ C:\Development\GitHub\Julia\Learning\20\_asm\analyze\_count\_char.jl:6 within `CountChar`

; Function Attrs: uwtable

define i64 @julia\_CountChar\_509({}\* noundef nonnull %0, i32 zeroext %1) #0 {

top:

%2 = **alloca** { i32, i64 }, align 8

%3 = **alloca** { i32, i64 }, align 8

; @ C:\Development\GitHub\Julia\Learning\20\_asm\analyze\_count\_char.jl:8 within `CountChar`

; ┌ @ strings/string.jl:400 within `iterate` @ strings/string.jl:400

; │┌ @ strings/string.jl:114 within `ncodeunits`

%4 = **bitcast** {}\* %0 to i64\*

%sizeof = **load** i64, i64\* %4, align 8

; │└

; │┌ @ int.jl:520 within `<` @ int.jl:513

%.not = **icmp** **eq** i64 %sizeof, 0

; │└

**br** i1 %.not, label %L112, label %L18

L18: ; preds = %top

; │ @ strings/string.jl:400 within `iterate` @ strings/string.jl:401

; │┌ @ strings/string.jl:120 within `codeunit`

; ││┌ @ strings/string.jl:112 within `pointer` @ strings/string.jl:111

; │││┌ @ pointer.jl:59 within `unsafe\_convert`

%5 = **bitcast** {}\* %0 to {}\*\*

%6 = **getelementptr** inbounds {}\*, {}\*\* %5, i64 1

; │││└

; │││ @ strings/string.jl:112 within `pointer`

; │││┌ @ pointer.jl:282 within `+`

%7 = **bitcast** {}\*\* %6 to i8\*

; ││└└

; ││┌ @ pointer.jl:119 within `unsafe\_load` @ pointer.jl:119

%pointerref = **load** i8, i8\* %7, align 1

; │└└

; │ @ strings/string.jl:400 within `iterate` @ strings/string.jl:402

; │┌ @ boot.jl:788 within `UInt32`

; ││┌ @ boot.jl:750 within `toUInt32`

%8 = **zext** i8 %pointerref to i32

; │└└

; │┌ @ int.jl:536 within `<<` @ int.jl:529

%9 = **shl** nuw i32 %8, 24

; │└

; │ @ strings/string.jl:400 within `iterate` @ strings/string.jl:403

; │┌ @ strings/string.jl:32 within `between`

; ││┌ @ bool.jl:38 within `&`

%10 = **icmp** **sgt** i8 %pointerref, -9

; │└└

**br** i1 %10, label %guard\_exit25, label %L37

L37: ; preds = %L18

; │ @ strings/string.jl:400 within `iterate` @ strings/string.jl:404

**call** void @j\_iterate\_continued\_511({ i32, i64 }\* noalias nocapture noundef nonnull sret({ i32, i64 }) %2, {}\* nonnull %0, i64 signext 1, i32 zeroext %9)

; └

; @ C:\Development\GitHub\Julia\Learning\20\_asm\analyze\_count\_char.jl:6 within `CountChar`

%unionalloca.sroa.0.0..sroa\_idx = **getelementptr** inbounds { i32, i64 }, { i32, i64 }\* %2, i64 0, i32 0

%unionalloca.sroa.0.0.copyload = **load** i32, i32\* %unionalloca.sroa.0.0..sroa\_idx, align 8

%unionalloca.sroa.8.0..sroa\_idx86 = **getelementptr** inbounds { i32, i64 }, { i32, i64 }\* %2, i64 0, i32 1

%unionalloca.sroa.8.0.copyload = **load** i64, i64\* %unionalloca.sroa.8.0..sroa\_idx86, align 8

; @ C:\Development\GitHub\Julia\Learning\20\_asm\analyze\_count\_char.jl:8 within `CountChar`

; ┌ @ strings/string.jl:400 within `iterate` @ strings/string.jl:404

**br** label %guard\_exit25

L80: ; preds = %guard\_exit31, %L80.lr.ph

%value\_phi3129 = **phi** i64 [ %16, %L80.lr.ph ], [ %value\_phi3, %guard\_exit31 ]

%.sroa.7.0128 = **phi** i64 [ %unionalloca.sroa.8.0.ph, %L80.lr.ph ], [ %unionalloca15.sroa.8.1.ph, %guard\_exit31 ]

; └

; @ C:\Development\GitHub\Julia\Learning\20\_asm\analyze\_count\_char.jl:12 within `CountChar`

; ┌ @ strings/string.jl:401 within `iterate`

; │┌ @ strings/string.jl:120 within `codeunit`

; ││┌ @ strings/string.jl:112 within `pointer`

; │││┌ @ pointer.jl:282 within `+`

%11 = **getelementptr** i8, i8\* %7, i64 -1

; │││└

; │││┌ @ pointer.jl:283 within `-`

%coercion11 = **getelementptr** i8, i8\* %11, i64 %.sroa.7.0128

; ││└└

; ││┌ @ pointer.jl:119 within `unsafe\_load` @ pointer.jl:119

%pointerref12 = **load** i8, i8\* %coercion11, align 1

; │└└

; │ @ strings/string.jl:402 within `iterate`

; │┌ @ boot.jl:788 within `UInt32`

; ││┌ @ boot.jl:750 within `toUInt32`

%12 = **zext** i8 %pointerref12 to i32

; │└└

; │┌ @ int.jl:536 within `<<` @ int.jl:529

%13 = **shl** nuw i32 %12, 24

; │└

; │ @ strings/string.jl:403 within `iterate`

; │┌ @ strings/string.jl:32 within `between`

; ││┌ @ bool.jl:38 within `&`

%14 = **icmp** **sgt** i8 %pointerref12, -9

; │└└

**br** i1 %14, label %L102, label %L100

L100: ; preds = %L80

; │ @ strings/string.jl:404 within `iterate`

**call** void @j\_iterate\_continued\_511({ i32, i64 }\* noalias nocapture noundef nonnull sret({ i32, i64 }) %3, {}\* nonnull %0, i64 signext %.sroa.7.0128, i32 zeroext %13)

; └

; @ C:\Development\GitHub\Julia\Learning\20\_asm\analyze\_count\_char.jl:6 within `CountChar`

%unionalloca15.sroa.0.0.copyload = **load** i32, i32\* %unionalloca15.sroa.0.0..sroa\_idx, align 8

%unionalloca15.sroa.8.0.copyload = **load** i64, i64\* %unionalloca15.sroa.8.0..sroa\_idx46, align 8

; @ C:\Development\GitHub\Julia\Learning\20\_asm\analyze\_count\_char.jl:12 within `CountChar`

; ┌ @ strings/string.jl:404 within `iterate`

**br** label %guard\_exit31

L102: ; preds = %L80

; │ @ strings/string.jl:403 within `iterate`

; │┌ @ int.jl:87 within `+`

%15 = **add** nuw nsw i64 %.sroa.7.0128, 1

; │└

**br** label %guard\_exit31

L112: ; preds = %guard\_exit31, %guard\_exit25, %top

%value\_phi18 = **phi** i64 [ 0, %top ], [ %16, %guard\_exit25 ], [ %value\_phi3, %guard\_exit31 ]

; └

; @ C:\Development\GitHub\Julia\Learning\20\_asm\analyze\_count\_char.jl:13 within `CountChar`

**ret** i64 %value\_phi18

guard\_exit25: ; preds = %L37, %L18

%unionalloca.sroa.0.0.ph = **phi** i32 [ %9, %L18 ], [ %unionalloca.sroa.0.0.copyload, %L37 ]

%unionalloca.sroa.8.0.ph = **phi** i64 [ 2, %L18 ], [ %unionalloca.sroa.8.0.copyload, %L37 ]

; @ C:\Development\GitHub\Julia\Learning\20\_asm\analyze\_count\_char.jl:9 within `CountChar`

; ┌ @ char.jl:213 within `==` @ promotion.jl:521

%.not105126 = **icmp** **eq** i32 %unionalloca.sroa.0.0.ph, %1

; └

%16 = **zext** i1 %.not105126 to i64

; @ C:\Development\GitHub\Julia\Learning\20\_asm\analyze\_count\_char.jl:12 within `CountChar`

; ┌ @ strings/string.jl:400 within `iterate`

; │┌ @ int.jl:1068 within `-` @ int.jl:86

%17 = **add** i64 %unionalloca.sroa.8.0.ph, -1

; │└

; │┌ @ int.jl:520 within `<` @ int.jl:513

%.not106127 = **icmp** **ult** i64 %17, %sizeof

; │└

**br** i1 %.not106127, label %L80.lr.ph, label %L112

L80.lr.ph: ; preds = %guard\_exit25

%unionalloca15.sroa.0.0..sroa\_idx = **getelementptr** inbounds { i32, i64 }, { i32, i64 }\* %3, i64 0, i32 0

%unionalloca15.sroa.8.0..sroa\_idx46 = **getelementptr** inbounds { i32, i64 }, { i32, i64 }\* %3, i64 0, i32 1

**br** label %L80

guard\_exit31: ; preds = %L102, %L100

%unionalloca15.sroa.8.1.ph = **phi** i64 [ %unionalloca15.sroa.8.0.copyload, %L100 ], [ %15, %L102 ]

%unionalloca15.sroa.0.1.ph = **phi** i32 [ %unionalloca15.sroa.0.0.copyload, %L100 ], [ %13, %L102 ]

; └

; @ C:\Development\GitHub\Julia\Learning\20\_asm\analyze\_count\_char.jl:9 within `CountChar`

; ┌ @ char.jl:213 within `==` @ promotion.jl:521

%.not105 = **icmp** **eq** i32 %unionalloca15.sroa.0.1.ph, %1

; └

%18 = **zext** i1 %.not105 to i64

%value\_phi3 = **add** i64 %value\_phi3129, %18

; @ C:\Development\GitHub\Julia\Learning\20\_asm\analyze\_count\_char.jl:12 within `CountChar`

; ┌ @ strings/string.jl:400 within `iterate`

; │┌ @ int.jl:1068 within `-` @ int.jl:86

%19 = **add** i64 %unionalloca15.sroa.8.1.ph, -1

; │└

; │┌ @ int.jl:520 within `<` @ int.jl:513

%.not106 = **icmp** **ult** i64 %19, %sizeof

; │└

**br** i1 %.not106, label %L80, label %L112

; └

}

--------- @code\_native CountChar

.text

.file "CountChar"

.globl julia\_CountChar\_534 # -- Begin function julia\_CountChar\_534

.p2align 4, 0x90

.type julia\_CountChar\_534,@function

julia\_CountChar\_534: # @julia\_CountChar\_534

; ┌ @ C:\Development\GitHub\Julia\Learning\20\_asm\analyze\_count\_char.jl:6 within `CountChar`

.cfi\_startproc

# %bb.0: # %top

**push** rbp

.cfi\_def\_cfa\_offset 16

.cfi\_offset rbp, -16

**mov** rbp, rsp

.cfi\_def\_cfa\_register rbp

; │ @ C:\Development\GitHub\Julia\Learning\20\_asm\analyze\_count\_char.jl:8 within `CountChar`

; │┌ @ strings/string.jl:400 within `iterate` @ strings/string.jl:400

; ││┌ @ strings/string.jl:114 within `ncodeunits`

**push** r15

**push** r14

**push** r12

**push** rsi

**push** rdi

**push** rbx

**sub** rsp, 64

.cfi\_offset rbx, -64

.cfi\_offset rdi, -56

.cfi\_offset rsi, -48

.cfi\_offset r12, -40

.cfi\_offset r14, -32

.cfi\_offset r15, -24

**mov** rsi, qword ptr [rcx]

; ││└

; ││┌ @ int.jl:520 within `<` @ int.jl:513

**test** rsi, rsi

; ││└

**je** .LBB0\_1

# %bb.2: # %L18

**mov** r15d, edx

**mov** rdi, rcx

; ││ @ strings/string.jl:400 within `iterate` @ strings/string.jl:402

; ││┌ @ boot.jl:788 within `UInt32`

; │││┌ @ boot.jl:750 within `toUInt32`

**movzx** eax, byte ptr [rcx + 8]

; ││└└

; ││┌ @ int.jl:536 within `<<` @ int.jl:529

**mov** r9d, eax

**shl** r9d, 24

**mov** r8d, 2

; ││└

; ││ @ strings/string.jl:400 within `iterate` @ strings/string.jl:403

; ││┌ @ strings/string.jl:32 within `between`

; │││┌ @ bool.jl:38 within `&`

**cmp** al, -9

; ││└└

**jg** .LBB0\_4

# %bb.3: # %L37

; ││ @ strings/string.jl:400 within `iterate` @ strings/string.jl:404

**movabs** rax, offset j\_iterate\_continued\_536

**lea** rcx, [rbp - 80]

**mov** r8d, 1

**mov** rdx, rdi

**call** rax

; │└

; │ @ C:\Development\GitHub\Julia\Learning\20\_asm\analyze\_count\_char.jl:6 within `CountChar`

**mov** r9d, dword ptr [rbp - 80]

**mov** r8, qword ptr [rbp - 72]

.LBB0\_4: # %guard\_exit25

; │ @ C:\Development\GitHub\Julia\Learning\20\_asm\analyze\_count\_char.jl:9 within `CountChar`

; │┌ @ char.jl:213 within `==` @ promotion.jl:521

**xor** ebx, ebx

**cmp** r9d, r15d

**sete** bl

; │└

; │ @ C:\Development\GitHub\Julia\Learning\20\_asm\analyze\_count\_char.jl:12 within `CountChar`

; │┌ @ strings/string.jl:400 within `iterate`

; ││┌ @ int.jl:1068 within `-` @ int.jl:86

**lea** rax, [r8 - 1]

; ││└

; ││┌ @ int.jl:520 within `<` @ int.jl:513

**cmp** rax, rsi

; ││└

**jae** .LBB0\_10

# %bb.5:

**movabs** r12, offset j\_iterate\_continued\_536

**lea** r14, [rbp - 64]

**jmp** .LBB0\_6

.p2align 4, 0x90

.LBB0\_8: # %L102

# in Loop: Header=BB0\_6 Depth=1

; ││ @ strings/string.jl:403 within `iterate`

; ││┌ @ int.jl:87 within `+`

**inc** r8

.LBB0\_9: # %guard\_exit31

# in Loop: Header=BB0\_6 Depth=1

; │└└

; │ @ C:\Development\GitHub\Julia\Learning\20\_asm\analyze\_count\_char.jl:9 within `CountChar`

; │┌ @ char.jl:213 within `==` @ promotion.jl:521

**xor** eax, eax

**cmp** r9d, r15d

**sete** al

; │└

**add** rbx, rax

; │ @ C:\Development\GitHub\Julia\Learning\20\_asm\analyze\_count\_char.jl:12 within `CountChar`

; │┌ @ strings/string.jl:400 within `iterate`

; ││┌ @ int.jl:1068 within `-` @ int.jl:86

**lea** rax, [r8 - 1]

; ││└

; ││┌ @ int.jl:520 within `<` @ int.jl:513

**cmp** rax, rsi

; ││└

**jae** .LBB0\_10

.LBB0\_6: # %L80

# =>This Inner Loop Header: Depth=1

; ││ @ strings/string.jl:402 within `iterate`

; ││┌ @ boot.jl:788 within `UInt32`

; │││┌ @ boot.jl:750 within `toUInt32`

**movzx** eax, byte ptr [rdi + r8 + 7]

; ││└└

; ││┌ @ int.jl:536 within `<<` @ int.jl:529

**mov** r9d, eax

**shl** r9d, 24

; ││└

; ││ @ strings/string.jl:403 within `iterate`

; ││┌ @ strings/string.jl:32 within `between`

; │││┌ @ bool.jl:38 within `&`

**cmp** al, -9

; ││└└

**jg** .LBB0\_8

# %bb.7: # %L100

# in Loop: Header=BB0\_6 Depth=1

; ││ @ strings/string.jl:404 within `iterate`

**mov** rcx, r14

**mov** rdx, rdi

**call** r12

; │└

; │ @ C:\Development\GitHub\Julia\Learning\20\_asm\analyze\_count\_char.jl:6 within `CountChar`

**mov** r9d, dword ptr [rbp - 64]

**mov** r8, qword ptr [rbp - 56]

; │ @ C:\Development\GitHub\Julia\Learning\20\_asm\analyze\_count\_char.jl:12 within `CountChar`

; │┌ @ strings/string.jl:404 within `iterate`

**jmp** .LBB0\_9

.LBB0\_1:

**xor** ebx, ebx

.LBB0\_10: # %L112

; │└

; │ @ C:\Development\GitHub\Julia\Learning\20\_asm\analyze\_count\_char.jl:13 within `CountChar`

**mov** rax, rbx

**add** rsp, 64

**pop** rbx

**pop** rdi

**pop** rsi

**pop** r12

**pop** r14

**pop** r15

**pop** rbp

**ret**

.Lfunc\_end0:

.size julia\_CountChar\_534, .Lfunc\_end0-julia\_CountChar\_534

.cfi\_endproc

; └

# -- End function

.type .L\_j\_const1,@object # @\_j\_const1

.section .rodata.cst8,"aM",@progbits,8

.p2align 3

.L\_j\_const1:

.quad 2 # 0x2

.size .L\_j\_const1, 8

.section ".note.GNU-stack","",@progbits

WOTAN\PierrC:\Development\GitHub\Julia\Learning\20\_asm