Bresia Prudente
bprude2

The generated assembly file is below. In the IA-32 code, the old fram pointer is saved and a new frame pointer is created in the bubble sort function. Additionally, 24 bytes are allocated from the stack while the literal 0 is moved into the 20 bytes above the base pointer.

However, the x86-64 code saves the old frame pointer into a different section of the code. The new frame pointer is created into another section as well.

Some sections of the x86-64 code use movq, addq, etc, as well as other registers (such as rax and rbp). These use 64 bit versions of the pointers, registers and variables so they have more memory within this code.

The same amount of memory isn't required for these codes. The IA-32 requires more memory than the x86-64. This is because x86-64 code is more compact, requiring fewer memory accesses and runs more efficiently than the other verison.

IA-32/x86-64 CODE

```
    .file  "bubble.c"
    .text
.globl bubble
    .type  bubble, @function
bubble:
.LFB2:
    pushq  %rbp
.LCFI0:
    movq   %rsp, %rbp
.LCFI1:
    movq   %rdi, -24(%rbp)
    movl   %esi, -28(%rbp)
    movl   $0, -12(%rbp)
    jmp    .L2
.L3:
    movl   $0, -8(%rbp)
    jmp    .L4
.L5:
    movl   -8(%rbp), %eax
    cltq
    salq   $2, %rax
    addq   -24(%rbp), %rax
    movl   (%rax), %ecx
    movq   -24(%rbp), %rdx
    addq   $4, %rdx
```

```
    movl   -8(%rbp), %eax
    cltq
    salq   $2, %rax
    leaq   (%rdx,%rax), %rax
    movl   (%rax), %eax
    cmpl   %eax, %ecx
    jle    .L6
    movl   -8(%rbp), %eax
    cltq
    salq   $2, %rax
    addq   -24(%rbp), %rax
    movl   (%rax), %eax
    movl   %eax, -4(%rbp)
    movl   -8(%rbp), %eax
    cltq
    salq   $2, %rax
    movq   %rax, %rcx
    addq   -24(%rbp), %rcx
    movq   -24(%rbp), %rdx
    addq   $4, %rdx
    movl   -8(%rbp), %eax
    cltq
    salq   $2, %rax
    leaq   (%rdx,%rax), %rax
    movl   (%rax), %eax
    movl   %eax, (%rcx)
    movq   -24(%rbp), %rdx
    addq   $4, %rdx
    movl   -8(%rbp), %eax
    cltq
    salq   $2, %rax
    addq   %rax, %rdx
    movl   -4(%rbp), %eax
    movl   %eax, (%rdx)
.L6:
    addl   $1, -8(%rbp)
.L4:
    movl   -12(%rbp), %edx
    movl   -28(%rbp), %eax
    subl   %edx, %eax
    subl   $1, %eax
    cmpl   -8(%rbp), %eax
    jg     .L5
    addl   $1, -12(%rbp)
.L2:
    movl   -28(%rbp), %eax
    subl   $1, %eax
    cmpl   -12(%rbp), %eax
    jg     .L3
```

```
    leave
    ret
.LFE2:
    .size  bubble, .-bubble
    .section    .rodata
.LC0:
    .string     "Enter number of elements"
.LC1:
    .string     "%d"
.LC2:
    .string     "Enter %d integers\n"
    .align 8
.LC3:
    .string     "Sorted list in ascending order:"
.LC4:
    .string     "%d\n"
    .text
.globl main
    .type  main, @function
main:
.LFB3:
    pushq  %rbp
.LCFI2:
    movq   %rsp, %rbp
.LCFI3:
    subq   $432, %rsp
.LCFI4:
    movl   $.LC0, %edi
    call   puts
    leaq   -420(%rbp), %rsi
    movl   $.LC1, %edi
    movl   $0, %eax
    call   scanf
    movl   -420(%rbp), %esi
    movl   $.LC2, %edi
    movl   $0, %eax
    call   printf
    movl   $0, -12(%rbp)
    jmp    .L12
.L13:
    movl   -12(%rbp), %eax
    cltq
    salq   $2, %rax
    leaq   -416(%rbp), %rsi
    addq   %rax, %rsi
    movl   $.LC1, %edi
    movl   $0, %eax
    call   scanf
    addl   $1, -12(%rbp)
```

```
.L12:
    movl   -420(%rbp), %eax
    cmpl   %eax, -12(%rbp)
    jl     .L13
    movl   -420(%rbp), %esi
    leaq   -416(%rbp), %rdi
    call   bubble
    movl   $.LC3, %edi
    call   puts
    movl   $0, -12(%rbp)
    jmp    .L15
.L16:
    movl   -12(%rbp), %eax
    cltq
    movl   -416(%rbp,%rax,4), %esi
    movl   $.LC4, %edi
    movl   $0, %eax
    call   printf
    addl   $1, -12(%rbp)
.L15:
    movl   -420(%rbp), %eax
    cmpl   %eax, -12(%rbp)
    jl     .L16
    movl   $0, %eax
    leave
    ret
.LFE3:
    .size  main, .-main
    .section    .eh_frame,"a",@progbits
.Lframe1:
    .long  .LECIE1-.LSCIE1
.LSCIE1:
    .long  0x0
    .byte  0x1
    .string     "zR"
    .uleb128 0x1
    .sleb128 -8
    .byte  0x10
    .uleb128 0x1
    .byte  0x3
    .byte  0xc
    .uleb128 0x7
    .uleb128 0x8
    .byte  0x90
    .uleb128 0x1
    .align 8
.LECIE1:
.LSFDE1:
    .long  .LEFDE1-.LASFDE1
```

```
.LASFDE1:
    .long  .LASFDE1-.Lframe1
    .long  .LFB2
    .long  .LFE2-.LFB2
    .uleb128 0x0
    .byte  0x4
    .long  .LCFI0-.LFB2
    .byte  0xe
    .uleb128 0x10
    .byte  0x86
    .uleb128 0x2
    .byte  0x4
    .long  .LCFI1-.LCFI0
    .byte  0xd
    .uleb128 0x6
    .align 8
.LEFDE1:
.LSFDE3:
    .long  .LEFDE3-.LASFDE3
.LASFDE3:
    .long  .LASFDE3-.Lframe1
    .long  .LFB3
    .long  .LFE3-.LFB3
    .uleb128 0x0
    .byte  0x4
    .long  .LCFI2-.LFB3
    .byte  0xe
    .uleb128 0x10
    .byte  0x86
    .uleb128 0x2
    .byte  0x4
    .long  .LCFI3-.LCFI2
    .byte  0xd
    .uleb128 0x6
    .align 8
.LEFDE3:
    .ident "GCC: (GNU) 4.1.2 20080704 (Red Hat 4.1.2-54)"
    .section    .note.GNU-stack,"",@progbits
```