Dcg87

Diff.txt contains side-by-side comparison of mystery.c (unoptimized vs. optimized).

Mystery

Basic functionality:

Computes the nth Fibonacci number.

Optimization:

Two things. First, uses a global array, *num*, to store previous calculations. Second, places -1 in *num[k]* to indicate that the kth Fibonacci number has not been calculated yet. This avoids calculating the same Fibonacci number twice.

Comparison: Un-optimized vs. Optimized

Note: un-optimized is on the left, optimized is on the right.

Main():

Dcg87

```
        .section        .rodata                 |       .section        .rodata.str1.1,"aMS",@progbits,1
.LC0:                                             .LC0:
        .string "Value: %d\n"                           .string "Value: %d\n"
        .text                                           .text
        .globl  main                                    .globl  main
        .type   main, @function                         .type   main, @function
main:                                             main:
.LFB2:                                          | .LFB27:
        .cfi_startproc                                  .cfi_startproc
        pushq   %rbp                            |       subq    $8, %rsp
        .cfi_def_cfa_offset 16                          .cfi_def_cfa_offset 16
        .cfi_offset 6, -16                      |       movq    8(%rsi), %rdi
        movq    %rsp, %rbp                      <
        .cfi_def_cfa_register 6                 <
        subq    $32, %rsp                       <
        movl    %edi, -20(%rbp)                 <
        movq    %rsi, -32(%rbp)                 <
        movq    -32(%rbp), %rax                 <
        addq    $8, %rax                        <
        movq    (%rax), %rax                    <
        movq    %rax, %rdi                      <
        movl    $0, %eax                                movl    $0, %eax
        call    atoi                                    call    atoi
        movl    %eax, -8(%rbp)                  |       movl    $num, %edx
        movl    $0, -4(%rbp)                    |       movl    $num+1600, %esi
        jmp     .L10                            |       movq    $-1, %rcx
.L11:                                           | .L11:
        movl    -4(%rbp), %eax                  |       movq    %rcx, (%rdx)
        cltq                                    |       addq    $8, %rdx
        movq    $-1, num(,%rax,8)               |       cmpq    %rsi, %rdx
        addl    $1, -4(%rbp)                    |       jne     .L11
.L10:                                           |       movslq  %eax, %rdi
        cmpl    $199, -4(%rbp)                  <
        jle     .L11                            <
        movl    -8(%rbp), %eax                  <
        cltq                                    <
        movq    %rax, %rdi                      <
        call    dothething                              call    dothething
        movq    %rax, %rsi                              movq    %rax, %rsi
        movl    $.LC0, %edi                             movl    $.LC0, %edi
        movl    $0, %eax                                movl    $0, %eax
        call    printf                                  call    printf
        movl    $0, %eax                                movl    $0, %eax
        leave                           |               addq    $8, %rsp
                                                                                100,2-9       91%
[1] 1:dcg87@factory:~/summer2017/pa2*                                   "factory.cs.rutgers.edu" 15:42 30-Jul-17
```

Prefers registers to using the stack (less overhead). Directly addresses *num* array instead of loading its address into *rax*.

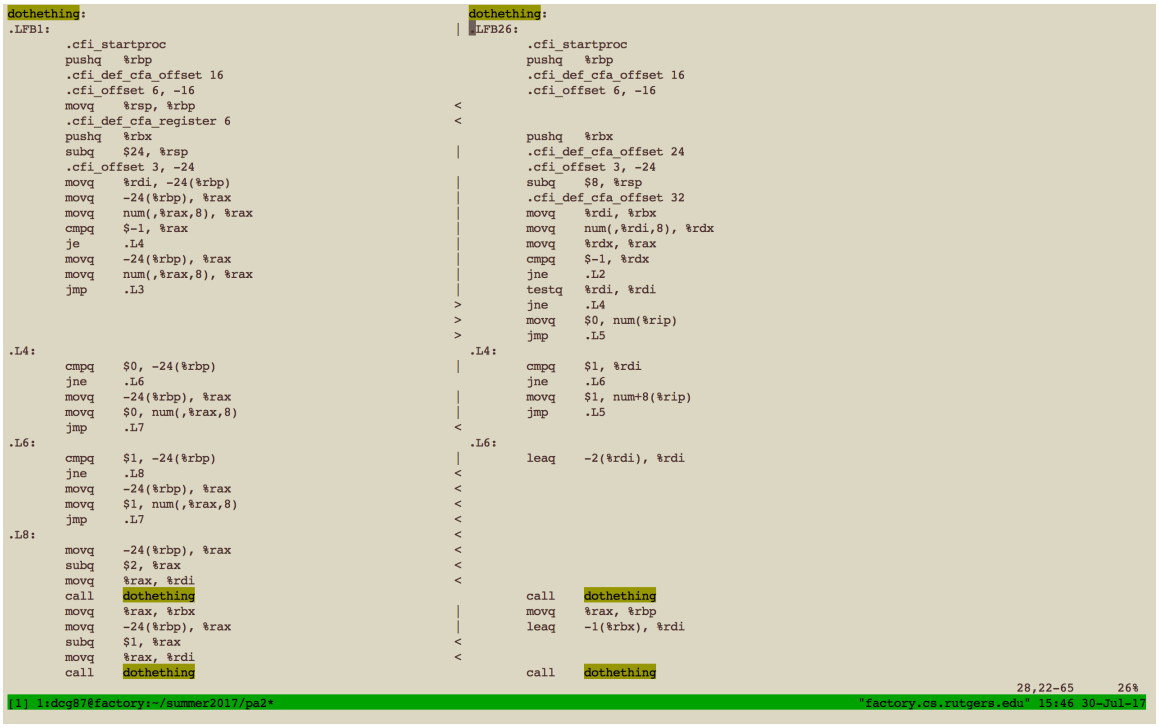
Add():

```
add:                                              add:
.LFB0:                                          | .LFB25:
        .cfi_startproc                                  .cfi_startproc
        pushq   %rbp                            |       leaq    (%rdi,%rsi), %rax
        .cfi_def_cfa_offset 16                  <
        .cfi_offset 6, -16                      <
        movq    %rsp, %rbp                      <
        .cfi_def_cfa_register 6                 <
        movq    %rdi, -8(%rbp)                  <
        movq    %rsi, -16(%rbp)                 <
        movq    -16(%rbp), %rax                 <
        movq    -8(%rbp), %rdx                  <
        addq    %rdx, %rax                      <
        popq    %rbp                            <
        .cfi_def_cfa 7, 8                       <
        ret                                             ret
        .cfi_endproc                                    .cfi_endproc
.LFE0:                                          | .LFE25:
        .size   add, .-add                              .size   add, .-add
        .globl  dothething                              .globl  dothething
        .type   dothething, @function                   .type   dothething, @function
dothething:                                       dothething:
.LFB1:                                          | .LFB26:
        .cfi_startproc                                  .cfi_startproc
        pushq   %rbp                                    pushq   %rbp
        .cfi_def_cfa_offset 16                          .cfi_def_cfa_offset 16
        .cfi_offset 6, -16                              .cfi_offset 6, -16
        movq    %rsp, %rbp                      <
        .cfi_def_cfa_register 6                 <
        pushq   %rbx                                    pushq   %rbx
        subq    $24, %rsp                       |       .cfi_def_cfa_offset 24
        .cfi_offset 3, -24                              .cfi_offset 3, -24
        movq    %rdi, -24(%rbp)                 |       subq    $8, %rsp
        movq    -24(%rbp), %rax                 |       .cfi_def_cfa_offset 32
        movq    num(,%rax,8), %rax              |       movq    %rdi, %rbx
        cmpq    $-1, %rax                       |       movq    num(,%rdi,8), %rdx
        je      .L4                             |       movq    %rdx, %rax
        movq    -24(%rbp), %rax                 |       cmpq    $-1, %rdx
        movq    num(,%rax,8), %rax              |       jne     .L2
        jmp     .L3                             |       testq   %rdi, %rdi
                                                >       jne     .L4
                                                >       movq    $0, num(%rip)
                                                >       jmp     .L5
.L4:                                              .L4:
                                                                                14,2-9        5%
[1] 1:dcg87@factory:~/summer2017/pa2*                                   "factory.cs.rutgers.edu" 15:45 30-Jul-17
```

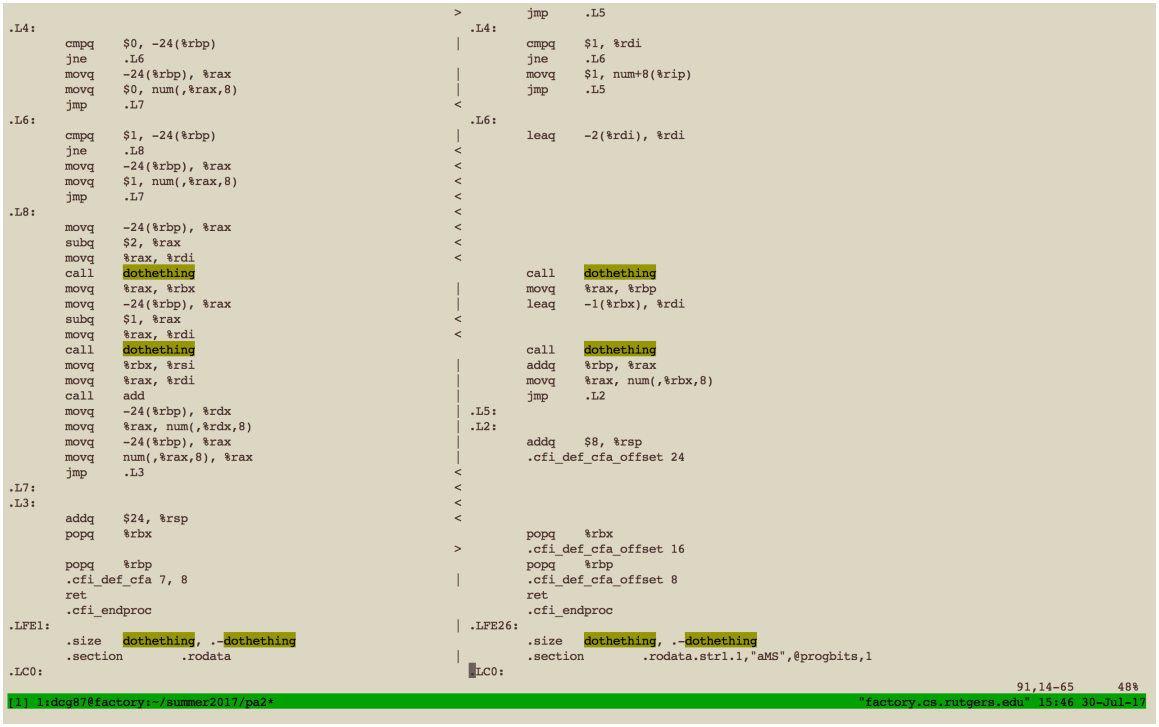
Dcg87

Prefers registers to using the stack (less overhead). Uses *leaq* for addition instead of copying registers. Dramatic decrease from 9 lines to 2 lines.

Dothething():



```
dothething:                                          dothething:
.LFB1:                                             | .LFB26:
        .cfi_startproc                                       .cfi_startproc
        pushq   %rbp                                         pushq   %rbp
        .cfi_def_cfa_offset 16                               .cfi_def_cfa_offset 16
        .cfi_offset 6, -16                                   .cfi_offset 6, -16
        movq    %rsp, %rbp                           <
        .cfi_def_cfa_register 6                      <
        pushq   %rbx                                         pushq   %rbx
        subq    $24, %rsp                            |       .cfi_def_cfa_offset 24
        .cfi_offset 3, -24                           |       .cfi_offset 3, -24
        movq    %rdi, -24(%rbp)                      |       subq    $8, %rsp
        movq    -24(%rbp), %rax                      |       .cfi_def_cfa_offset 32
        movq    num(,%rax,8), %rax                   |       movq    %rdi, %rbx
        cmpq    $-1, %rax                            |       movq    num(,%rdi,8), %rdx
        je      .L4                                  |       movq    %rdx, %rax
        movq    -24(%rbp), %rax                      |       cmpq    $-1, %rdx
        movq    num(,%rax,8), %rax                   |       jne     .L2
        jmp     .L3                                  |       testq   %rdi, %rdi
                                                     >       jne     .L4
                                                     >       movq    $0, num(%rip)
                                                     >       jmp     .L5
.L4:                                                   .L4:
        cmpq    $0, -24(%rbp)                        |       cmpq    $1, %rdi
        jne     .L6                                          jne     .L6
        movq    -24(%rbp), %rax                      |       movq    $1, num+8(%rip)
        movq    $0, num(,%rax,8)                     |       jmp     .L5
        jmp     .L7                                  <
.L6:                                                   .L6:
        cmpq    $1, -24(%rbp)                        |       leaq    -2(%rdi), %rdi
        jne     .L8                                  <
        movq    -24(%rbp), %rax                      <
        movq    $1, num(,%rax,8)                     <
        jmp     .L7                                  <
.L8:                                                 <
        movq    -24(%rbp), %rax                      <
        subq    $2, %rax                             <
        movq    %rax, %rdi                           <
        call    dothething                                   call    dothething
        movq    %rax, %rbx                           |       movq    %rax, %rbp
        movq    -24(%rbp), %rax                      |       leaq    -1(%rbx), %rdi
        subq    $1, %rax                             <
        movq    %rax, %rdi                           <
        call    dothething                                   call    dothething
                                                                                    28,22-65      26%
[1] 1:dcg87@factory:~/summer2017/pa2*                                "factory.cs.rutgers.edu" 15:46 30-Jul-17
```



Prefers registers to using the stack (less overhead). One exception is *push rbx*.

Formula

Design:

Used a *struct* to represent each term in the expansion. Three fields: *coefficient*, *variable*, *exponent*. Iterate for each power, calculate nCr. nCr() is by far most expensive operation in loop, so efficiency is limited by that of nCr().