Assembly Programming

Naeem Hossain

November 6, 2016

Computer Architecture

This assignment is designed to give you additional practice in reading and writing Assembly Language programs. As discussed in lecture, unless you are working in increasingly rare areas such as low-level OS development, you are unlikely to be reading and/or writing Assembly Language programs in the remainder of your career. However, we are still requiring you to read and write some here to make sure you understand the computing model underlying your C and Java programs. In addition, being able to read Assembly Language is particularly important because there are times when you need to understand what the compiler is doing to your code.

There are two parts. In the first part, you will write two small functions in Assembly. In the second part, you will be deciphering (that is, write equivalent C code) an Assembly Language program. You will also be asked to compare unoptimized and optimized versions of the code and explained what the compiler did when it optimized the code.

Important: On most of the iLab machines that we checked, e.g., several of the Design Pattern machines, including adapter.cs.rutgers.edu, command.cs.rutgers.edu, and factory.cs.rutgers.edu, gcc by default will generate x86 Assembly, which is what we want. Do be careful and check the generated assembly, though, in case some of the machines are 64-bit processors, with gcc configured to generate x86-64 code. If you see Assembly code with registers beginning with %r (e.g., %rbx or %r9), then it is x86-64 and not what you want. Move to another iLab machine.

Part 2: Reading x86 Assembly

In this part, you are asked to decipher the Assembly Language program in the attached mystery.s file. Specifically, you need to provide a concise description of what the program does and how it does it. You should also implement a C program mystery that performs the same task in the same manner that the code in the attached file mystery.s does. The provided program takes a single integer as input.

$ gcc -m32 -o mystery mystery.s $ ./mystery 41 Value: 165580141

Hint: This program performs a well known and easily recognizable computation. However, it includes an optimization to speedup the computation. You need to figure out both the basic functionality as well as the optimization, describe them, and replicate them in your C code.

Another Hint: You are not strictly required to go backward from the mystery.s file that we give you. That is, when you start writing your mystery.c program, you can compile it to Assembly (gcc -S), and compare the generated code against our mystery.s. Our mystery.s was generated on factory.cs.rutgers.edu so you should be able to generate the exact same code. Once you have implemented your C program, you should compile it with and without the -O option (optimization). You should then compare the two versions and explain the differences inside the mystery function.

Analysis:

I figured out that this program implemented the FIbonacci Sequence on numbers. Two things gave this away. First, the double call to “dothething” gave away a recursive call, so recursive algorithms immediately came to mind. Second, the constant comparisons between elements %ebp and %eax, especially the mathematical operations called in the “add” and “leal” functions made me think it was the Fibonacci Sequence. Finally, my suspicions were confirmed when I tested the mystery value and found that the FIbonacci Sequence’s 41st element is 165580141.

I ran

gcc -O2 -S -c mystery.c

And found that my C code had been optimized by shaving the amount of operations in certain registers, like %eax and %ebp. I think the compiler made these changes to make the time/space of the program more efficient.

Assembly is actually far easier to read than it is to write, since each minute possible action must be manually performed. However, the comparative level of freedom that Assembly gives is pretty unmatched by higher-level languages.

USAGE:

./mystery <(positive int < 46)>

Comparison:

```
bash-4.2$ diff --side-by-side mystery.unoptimized.s mystery.s
        .file   "mystery.c"                                     .file   "mystery.c"
        .comm num,796,32                                <
        .text                                                   .text
        .globl  add                                             .globl  add
        .type   add, @function                                          .type   add,
@function
add:                                                    add:
```

```
.LFB2:                                  |   .LFB37:
        .cfi_startproc                          .cfi_startproc
        pushq  %rbp                     |       leal     (%rdi,%rsi), %eax
        .cfi_def_cfa_offset 16                  <
        .cfi_offset 6, -16              <
        movq   %rsp, %rbp               <
        .cfi_def_cfa_register 6                 <
        movl   %edi, -4(%rbp)                   <
        movl   %esi, -8(%rbp)                   <
        movl   -8(%rbp), %eax                   <
        movl   -4(%rbp), %edx                   <
        addl   %edx, %eax               <
        popq   %rbp                     <
        .cfi_def_cfa 7, 8               <
        ret                                     ret
        .cfi_endproc                            .cfi_endproc
.LFE2:                                  |   .LFE37:
        .size   add, .-add                      .size   add, .-add
        .globl  dothething                      .globl  dothething
        .type   dothething, @function                   .type   dothething,
@function
dothething:                                 dothething:
.LFB3:                                  |   .LFB38:
        .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
        movl    %edi, -20(%rbp)                 |       subq    $8, %rsp
        cmpl    $0, -20(%rbp)                   |       .cfi_def_cfa_offset 32
        jg      .L4                             |       movl    %edi, %ebx
                                                >       testl   %edi, %edi
                                                >       jle     .L4
                                                >       cmpl    $1, %edi
                                                >       je      .L5
        movl    $0, %eax                                movl    $0, %eax
        jmp     .L5                             |       cmpl    $1, %edi
.L4:                                            |       jle     .L3
        cmpl    $1, -20(%rbp)                   |       leal    -2(%rdi), %edi
        jne     .L6                             <
        movl    $1, %eax                        <
        jmp     .L5                             <
.L6:                                            <
        cmpl    $1, -20(%rbp)                   <
        jle     .L7                             <
        movl    -20(%rbp), %eax                 <
        subl    $2, %eax                        <
        movl    %eax, %edi                      <
        call    dothething                              call    dothething
        movl    %eax, %ebx                      |       movl    %eax, %ebp
        movl    -20(%rbp), %eax                 |       leal    -1(%rbx),
%edi
        subl    $1, %eax                        <
        movl    %eax, %edi                      <
        call    dothething                              call    dothething
        movl    %ebx, %esi                      |       addl    %ebp, %eax
        movl    %eax, %edi                      |       jmp     .L3
        call    add                             |   .L4:
        jmp     .L5                             <
```

```
.L7:                                                    <
        movl    $0, %eax                                        movl    $0, %eax
                                                        >       jmp     .L3
.L5:                                                        .L5:
        addq    $24, %rsp                               |       movl    $1, %eax
                                                        >   .L3:
                                                        >       addq    $8, %rsp
                                                        >       .cfi_def_cfa_offset 24
        popq    %rbx                                            popq    %rbx
                                                        >       .cfi_def_cfa_offset 16
        popq    %rbp                                            popq    %rbp
        .cfi_def_cfa 7, 8                               |       .cfi_def_cfa_offset 8
        ret                                                     ret
        .cfi_endproc                                            .cfi_endproc
.LFE3:                                                  |   .LFE38:
        .size   dothething, .-dothething                        .size   dothething,
.-dothething
        .section        .rodata                                 |
.section.rodata.str1.1,"aMS",@progbits,1
.LC0:                                                       .LC0:
        .string "Incorrect number of arguments"                         .string"Incorrect
number of arguments"
                                                        >
.section.rodata.str1.8,"aMS",@progbits,1
        .align 8                                                .align 8
.LC1:                                                       .LC1:
        .string "Integer size is too small. Range is 1 -> 46"           .string"Integer size is too
small. Range is 1 -> 46"
        .align 8                                                .align 8
.LC2:                                                       .LC2:
        .string "Integer size is too large. Range is 1 -> 46"           .string"Integer size is too
large. Range is 1 -> 46"
                                                        >       .section.rodata.str1.1
```

```
.LC3:                                                   .LC3:
        .string "Value: "                                       .string"Value: "
.LC4:                                                   .LC4:
        .string "%s"                                            .string"%s"
.LC5:                                                   .LC5:
        .string "%d\n"                                                  .string"%d\n"
        .text                                                   .text
        .globl  main                                            .globl  main
        .type   main, @function                                         .type   main,
@function                                               @function
main:                                                   main:
.LFB4:                                                  .LFB39:
        .cfi_startproc                                          .cfi_startproc
        pushq   %rbp                                            pushq   %rbp
        .cfi_def_cfa_offset 16                                          .cfi_def_cfa_offset
16                                                      16
        .cfi_offset 6, -16                                      .cfi_offset 6, -16
        movq    %rsp, %rbp                              |               pushq   %rbx
        .cfi_def_cfa_register 6                         |               .cfi_def_cfa_offset
24                                                      |
        subq    $32, %rsp                               |               .cfi_offset 3, -24
        movl    %edi, -20(%rbp)                         |               subq    $8, %rsp
        movq    %rsi, -32(%rbp)                         |               .cfi_def_cfa_offset
32                                                      |
        movl    $0, -4(%rbp)                            |               movl    %edi, %ebp
        movq    -32(%rbp), %rax                         |               movq    8(%rsi), %rdi
        addq    $8, %rax                                |               movl    $10, %edx
        movq    (%rax), %rax                            |               movl    $0, %esi
        movq    %rax, %rdi                              |               call    strtol
        call    atoi                                    |               cmpl    $2, %ebp
        movl    %eax, -8(%rbp)                          <
        cmpl    $2, -20(%rbp)                           <
        jle     .L9                                                     jle     .L9
```

```
        movl    $.LC0, %edi
        call    puts
        movl    $-1, %eax
        jmp     .L10
.L9:
        cmpl    $0, -20(%rbp)
        jg      .L11
        movl    $.LC0, %edi
        call    puts
        movl    $-1, %eax
        jmp     .L10
.L11:
        cmpl    $0, -8(%rbp)
        jns     .L12
        movl    $.LC1, %edi
        call    puts
        movl    $-1, %eax
        jmp     .L10
.L12:
        cmpl    $46, -8(%rbp)
        jle     .L13
        movl    $.LC2, %edi
        call    puts
        movl    $-1, %eax
        jmp     .L10
.L13:
        jmp     .L14
.L15:
        movl    -4(%rbp), %eax
        cltq
```

```
        movl    $.LC0, %edi
        call    puts
        movl    $-1, %eax
        jmp     .L10
.L9:
|       movq    %rax, %rbx
>       testl   %ebp, %ebp
        jg      .L11
        movl    $.LC0, %edi
        call    puts
        movl    $-1, %eax
        jmp     .L10
.L11:
|       testl   %eax, %eax
        jns     .L12
        movl    $.LC1, %edi
        call    puts
        movl    $-1, %eax
        jmp     .L10
.L12:
|       cmpl    $46, %eax
        jle     .L13
        movl    $.LC2, %edi
        call    puts
        movl    $-1, %eax
        jmp     .L10
.L13:
|       movl    $num, %edx
>       movl    $num+796, %ecx
.L15:
  |     movl    $-1, (%rdx)
|       addq    $4, %rdx
```

```
        movl    $-1, num(,%rax,4)                |               cmpq    %rcx, %rdx
        addl    $1, -4(%rbp)                     |               jne     .L15
.L14:                                            <
        cmpl    $198, -4(%rbp)                           <
        jle     .L15                             <
        movl    $.LC3, %esi                              movl    $.LC3, %esi
        movl    $.LC4, %edi                              movl    $.LC4, %edi
        movl    $0, %eax                                 movl    $0, %eax
        call    printf                                   call    printf
        movl    -8(%rbp), %eax                           |       movl    %ebx, %edi
        movl    %eax, %edi                       <
        call    dothething                               call    dothething
        movl    %eax, %esi                               movl    %eax, %esi
        movl    $.LC5, %edi                              movl    $.LC5, %edi
        movl    $0, %eax                                 movl    $0, %eax
        call    printf                                   call    printf
        movl    $0, %eax                                 movl    $0, %eax
.L10:                                                .L10:
        leave                                    |               addq    $8, %rsp
        .cfi_def_cfa 7, 8                        |               .cfi_def_cfa_offset 24
                                                 >               popq    %rbx
                                                 >               .cfi_def_cfa_offset 16
                                                 >               popq    %rbp
                                                 >               .cfi_def_cfa_offset 8
        ret                                                      ret
        .cfi_endproc                                             .cfi_endproc
.LFE4:                                           |           .LFE39:
        .size   main, .-main                                     .size   main, .-main
                                                 >               .comm num,796,32
        .ident  "GCC: (GNU) 4.8.5 20150623 (Red Hat 4.8.5-4)"            .ident  "GCC: (GNU)
4.8.5 20150623 (Red Hat 4.8.5-4)"
        .section        .note.GNU-stack,"",@progbits
```

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