Heimadæmi 7

sbb51@hi.is

October 2022

Dæmi 1

Assembly

```
hoho:
     leaq    31(%rdi), %rax   // Erum að setja %rax sem %rdi+31
     testq   %rdi, %rdi       // Skoða hvort %rdi er stærra en 0
     cmovns  %rdi, %rax       // Setjum %rdi sem %rax ef testq >= 0
     sarq    $5, %rax         // Shiftum 5 til hægri um %rax bita
     ret
```

Samsvarandi C-forrit

```c
long hoho(long x){
    long t1 = 31+x;
    if(x > 0) t1 = x;
    t1 >>= 5;
    return t1;
}
```

Dæmi 2

Assembly

```
    whi:
        movl    $0, %eax          // Núlstillum %eax
        jmp     .L2               // Hoppum yfir í .L2
    .L3:
        leaq    -1(%rdi), %rdx    // Setja %rdx sem %rdi-1
        andq    %rdx, %rdi        // Setja %rdi sem %rdx og %rdi og-að saman
        xorq    $1, %rax          // Setja %rax sem 1 og %rax xor-að saman
    .L2:
        testq   %rdi, %rdi        // Tékkar hvort %rdi <= 0
        jne     .L3               // Fer í .L3 ef %rdi er ekki núll
        ret
```

Samsvarandi C-forrit

```
long whi(long k) {
    long t1 = 0;              // movl    \$0, \%eax
    while(k!=0){              // testq   \%rdi, \%rdi
        long t2 = k - 1;      // leaq    -1(\%rdi), \%rdx
        k = k & t2;           // andq    \%rdx, \%rdi
        t1 = t1 ^ 1;          // xorq    \$1, \%rax
    }
    return t1;
}
```

Dæmi 3

a)

```
int func(int n){
    if (n <= 1)                         .L3
        return 1;
    return (n * 3) + func(n / 4);
}
```

```
int func(int n){
    if (n <= 1)
    // cmpl     \$1, \%edi  - Setjum flögg á \%edi
    // jle      .L3         - Skoðum hvort \%edi er tómt ef svo er hoppum yfir í .L3

        return 1;
        // movl     \$1, \%eax    - Setjum \%eax sem 1

    return (n * 3) + func(n / 4);
    // leal (\%rdi,\%rdi,2), \%ebx - Setjum \%ebx sem \%rdi*3
    // leal    3(\%rdi), \%eax     - Setjum \%eax sem \%rdi+3
    // testl   \%edi, \%edi        _ Tékkar hvort \%edi <= 0
    // cmovs   \%eax, \%edi        - Setur \%edi inn í \%edx ef \%edi >= 0
    // sarl    \$2, \%edi          _ Setur \%edx sem \%edx / 4
    // movl    \%edx, \%edi        - Setur \%edx inn í \%edi
    // call    func                - Recursion call
    // addl    \%ebx, \%eax        - Leggur saman \%ebx og \%eax
```

Dæmi 4

a)

```
fact:
    cmpq    $1, %rdi          // Skoðum hvort %rdi er <= 1
    jg      .L8               // Hoppum yfir í .L8
    movl    $1, %eax          // Setjum %eax sem 1
    ret
.L8:
    pushq   %rbx              // Push %rbx
    movq    %rdi, %rbx        // Setjum %rbx sem %rdi
    leaq    -1(%rdi), %rdi    // Setjum %rdi sem %rdi - 1
    call    fact              //
    imulq   %rbx, %rax        // Setjum %rax sem %rax * %rbx
    popq    %rbx              // Pop %rbx
    ret
```

Við notum popq og pushq til að passa uppá %rbx. Við pushum %rbx þar sem við erum að setja það inní %rdi og þurfum að geyma það einhversstaðar. Popum því síðan í lokin þar sem við þurfum ekki að geyma það lengur og skilum því að lokum.

b)

```
fact:
    movl    $1, %eax          // Setjum %eax sem 1
    cmpq    $1, %rdi          // Skoðum hvort %rdi er <= 1
    jle     .L1               // Skoðum hvort \%rdi er tómt ef svo er hoppum yfir í .L1
.L2:
    movq    %rdi, %rdx        // Setjum %rdx sem %rdi
    subq    $1, %rdi          // Setjum %rdi sem %rdi - 1
    imulq   %rdx, %rax        // Setjum %rax sem %rax * %rbx
    cmpq    $1, %rdi          // Skoðum hvort %rdi er <= 1
    jne     .L2               // Hoppum í L2
.L1:
    ret
```

Það er hraðvirkara þar sem við þurfum ekki að push-a og pop-a.

Dæmi 5

```
int sswitch(int x) {
    int t1;
    switch (x)
    {
        case 0:
        case 2:
            t1 = 2*x + 3;
            break;
            // Skv L4 í assembly kóðanum þá eru case 0 og 2 bæði L6.
        case 3:
        case 4:
            t1 = x + 3;
            break;
            // Skv L4 í assembly kóðanum þá eru case 3 og 4 bæði L5.
        case 5:
            int t2 = x*8;
            t1 = 3;
            t1 = t1 - t2;
            break;
            // Skv L4 í assembly kóðanum þá er case 5 L3.
        default:
            t1 = x;
            break;
            // Skv L4 í assembly kóðanum þá er case 1 L7 eins og default case-ið okkar.
    }
    return t1;
}
```