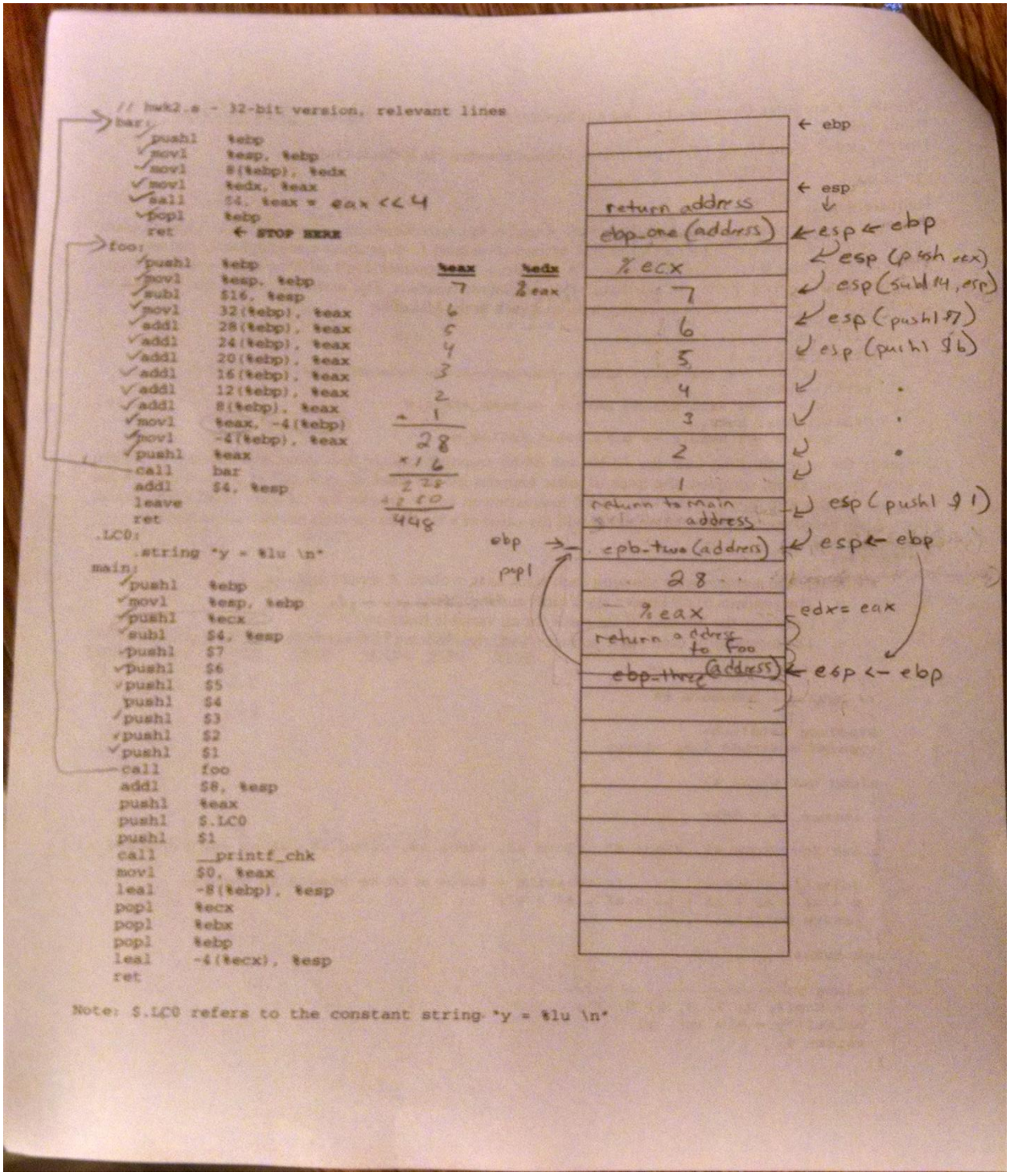
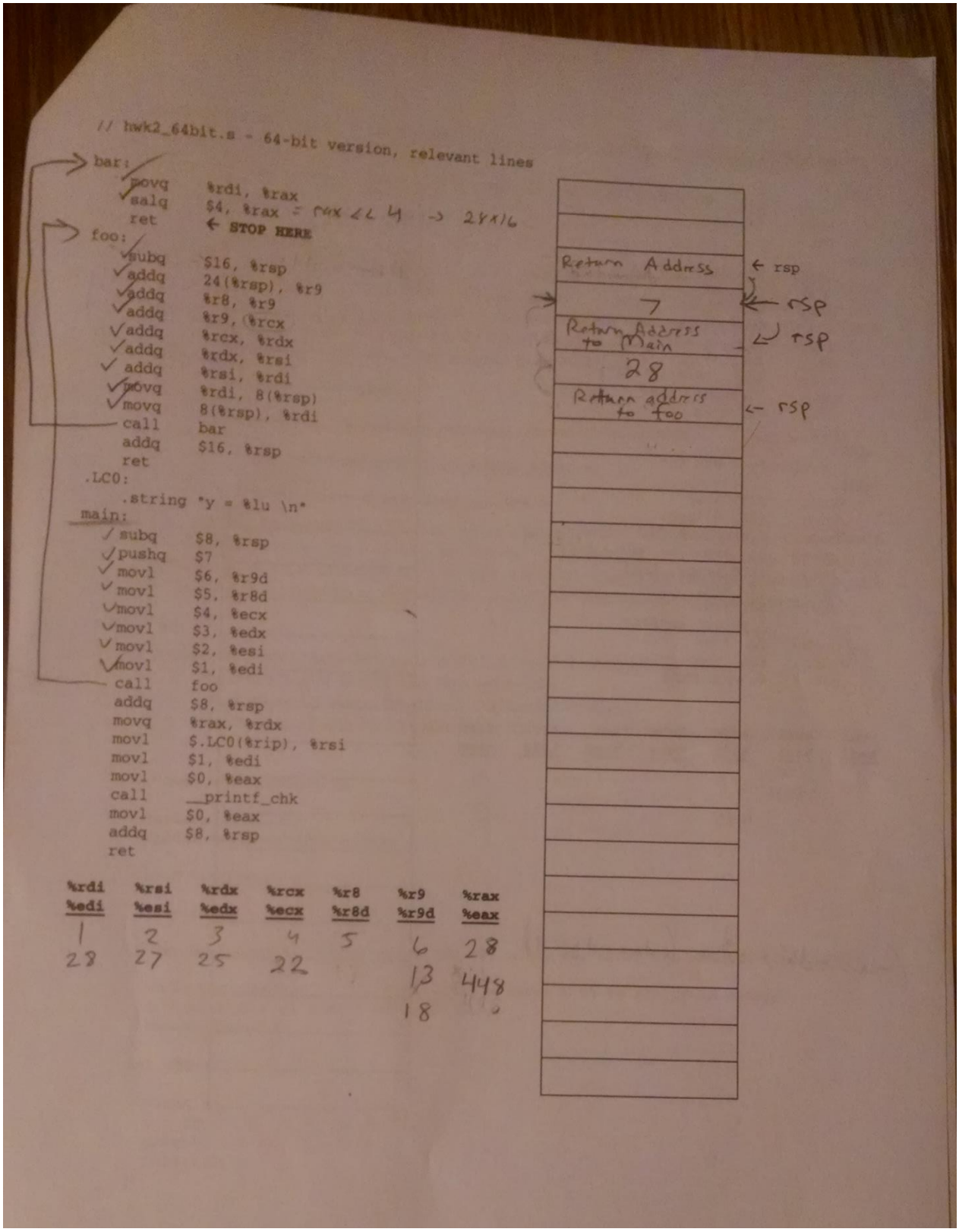


```
// hwk2.s - 32-bit version, relevant lines
bar:
    pushl   %ebp
    movl    %esp, %ebp
    movl    8(%ebp), %edx
    movl    %edx, %eax
    sall    $4, %eax = eax << 4
    popl    %ebp
    ret     <- STOP HERE
foo:
    pushl   %ebp                        %eax    %edx
    movl    %esp, %ebp                  7       %eax
    subl    $16, %esp                   6
    movl    32(%ebp), %eax              5
    addl    28(%ebp), %eax              4
    addl    24(%ebp), %eax              3
    addl    20(%ebp), %eax              2
    addl    16(%ebp), %eax              + 1
    addl    12(%ebp), %eax              28
    addl    8(%ebp), %eax               x 16
    movl    %eax, -4(%ebp)              168
    movl    -4(%ebp), %eax              +280
    pushl   %eax                        448
    call    bar
    addl    $4, %esp
    leave
    ret
.LC0:
    .string "y = %lu \n"
main:
    pushl   %ebp
    movl    %esp, %ebp
    pushl   %ecx
    subl    $4, %esp
    pushl   $7
    pushl   $6
    pushl   $5
    pushl   $4
    pushl   $3
    pushl   $2
    pushl   $1
    call    foo
    addl    $8, %esp
    pushl   %eax
    pushl   $.LC0
    pushl   $1
    call    __printf_chk
    movl    $0, %eax
    leal    -8(%ebp), %esp
    popl    %ecx
    popl    %ebx
    popl    %ebp
    leal    -4(%ecx), %esp
    ret

Note: $.LC0 refers to the constant string "y = %lu \n"
```



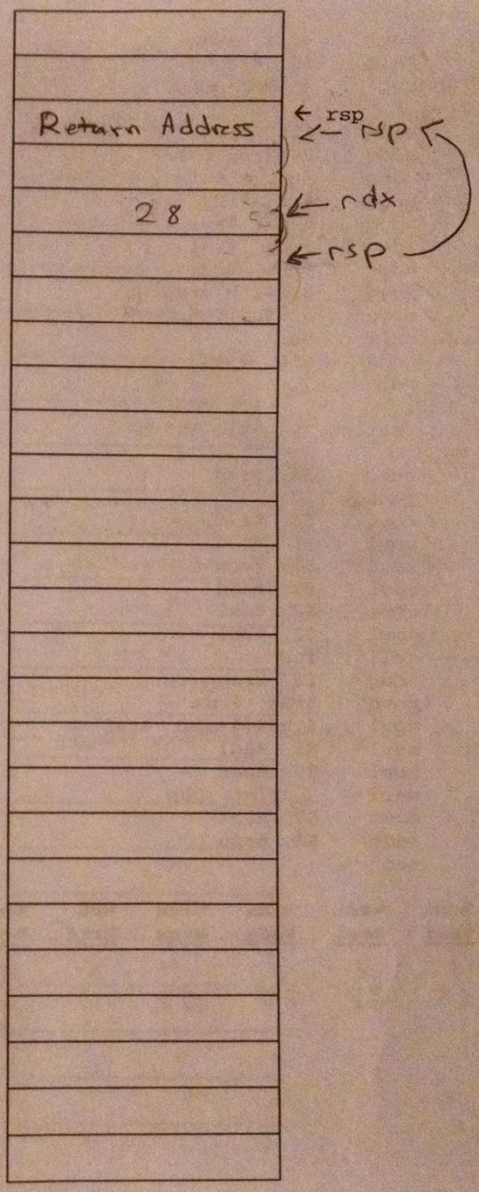
```
// hwk2_64bit.s - 64-bit version, relevant lines
bar:
    movq    %rdi, %rax
    salq    $4, %rax = rax << 4 -> 28*16
    ret             ← STOP HERE
foo:
    subq    $16, %rsp
    addq    24(%rsp), %r9
    addq    %r8, %r9
    addq    %r9, %rcx
    addq    %rcx, %rdx
    addq    %rdx, %rsi
    addq    %rsi, %rdi
    movq    %rdi, 8(%rsp)
    movq    8(%rsp), %rdi
    call    bar
    addq    $16, %rsp
    ret
.LC0:
    .string "y = %lu \n"
main:
    subq    $8, %rsp
    pushq   $7
    movl    $6, %r9d
    movl    $5, %r8d
    movl    $4, %ecx
    movl    $3, %edx
    movl    $2, %esi
    movl    $1, %edi
    call    foo
    addq    $8, %rsp
    movq    %rax, %rdx
    movl    $.LC0(%rip), %rsi
    movl    $1, %edi
    movl    $0, %eax
    call    __printf_chk
    movl    $0, %eax
    addq    $8, %rsp
    ret
```


%rdi	%rsi	%rdx	%rcx	%r8	%r9	%rax
%edi	%esi	%edx	%ecx	%r8d	%r9d	%eax
1	2	3	4	5	6	28
28	27	25	22		13	448
					18	

```
// hwk2_64bit_inline.s - 64-bit version, inline functions allowed, relevant lines

bar:
    movq    %rdi, %rax
    salq    $4, %rax
    ret
foo:
    addq    8(%rsp), %r9
    addq    %r8, %r9
    addq    %r9, %rcx
    addq    %rcx, %rdx
    addq    %rdx, %rsi
    addq    %rsi, %rdi
    movq    %rdi, -8(%rsp)
    movq    -8(%rsp), %rax
    salq    $4, %rax
    ret
.LC0:
    .string "y = %lu \n"
main:
    subq    $24, %rsp
    movq    $28, 8(%rsp)
    movq    8(%rsp), %rdx
    salq    $4, %rdx -> rdx = rdx<<4
    leaq    .LC0(%rip), %rsi
    movl    $1, %edi
    movl    $0, %eax
    call    __printf_chk@PLT
    movl    $0, %eax
    addq    $24, %rsp
    ret     <- STOP HERE
```

%rdi	%rsi	%rdx	%rcx	%r8	%r9	%rax
%edi	%esi	%edx	%ecx	%r8d	%r9d	%eax
1	.LC0's address	28				0
		448				0



$(rdx = rdx * 2^4) \approx (rdx = rdx << 4)$