CS61 Lecture 9

October 1, 2019

Ex Program 1:

specifies 8-byte

parameter registers, rdi comes first

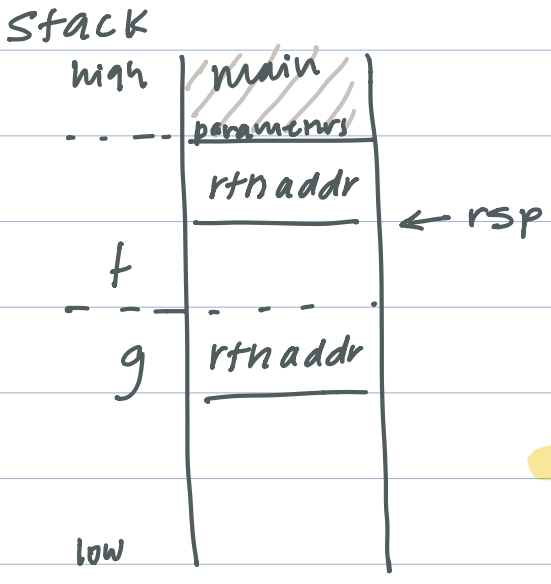
```
subq  %rsi, %rdi      rdi - rsi → rdi
leaq  (%rdi, %rdx), %rax      rdi + rdx → rax
```

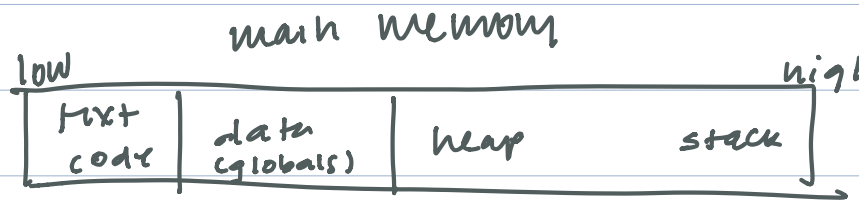
%rax ← return register

mangler: distinguishes functions w/ different parameter types

inputs to functions stored in:

edi, esi, edx, ecx, r8d, r9d, and then stack





rip ≡ instruction pointer

ALIGNMENT

all stack frames are 16b aligned

Ex Program 2:

```
subq   $8, %rsp
call   _Z1gi@PLT
addq   $8, %rsp
addl   $1, %eax
ret
```

Control Flow

1. sequential execution
2. unconditional jump
   ex: ret
3. conditional jump

cmp = compare (subtract)

condition code reg (1 bit)
has a bunch of flags

zf ≡ zero
sf ≡ sign
of ≡ overflow
cf ≡ carry

EX Program 3:

```
.LFB0
        cmpl    %edx, %esi      # c, b
        jg      .L3             # if (c == b)
        movl    %esi, %eax
        ret                     # return b
.L3
        movl    %edi, %eax      # a
        ret                     # return a
```

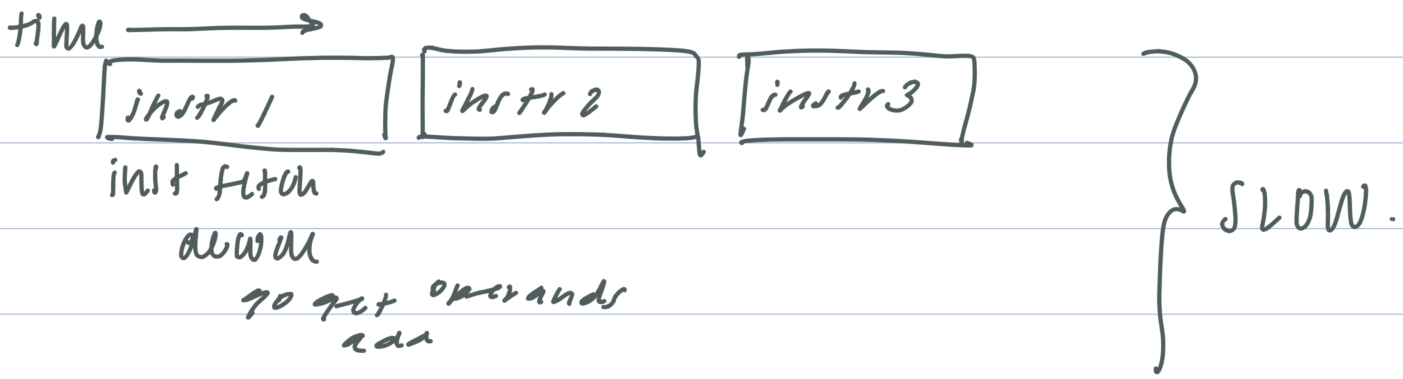
// jl .LE if (b < c)

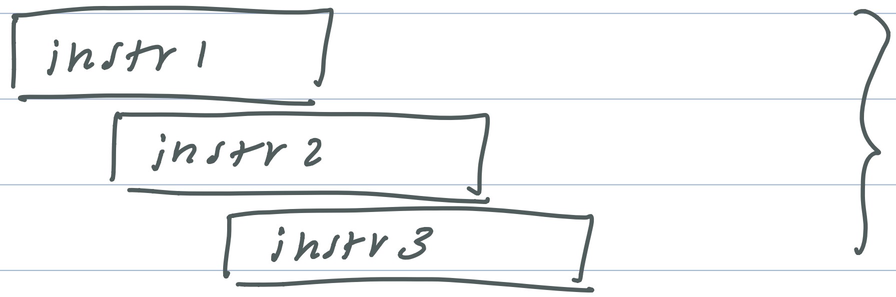
EX Program 4:

```
        cmpl    %esi, %edi
        movl    $0, %eax
        cmovge  %edi, %eax      # move if greater than
```

My new life motto: "Your intuition will tell you something and your intuition is wrong."

- Michael Smith





pipelining, which conditional jumps cannot do.

EX Program 5

similar to cmp, except uses &

```
.LFB0
    testl   %esi, %esi
    jle     .L4
    movq    %rdi, %rdx
    leal    -1(%rsi), %eax
    leaq    4(%rdi,%rax,4), %rsi
    movl    $0, %eax
.L3
    movslq  (%rdx), %rcx
    addq    %rcx, %rax
    addq    $4, %rdx
    cmpq    %rsi, %rdx
    jne     .L3
    rep     ret
```

move value of rdx to rcx

```
.L4
    movl    $0, %eax
    ret
```