CS61 Lecture 8 — September 26

Logistics

- reqvad 1 problem set

Review of last time

		src, dest
· Data movement	movl	$100, %eax
		a(%rip), %eax

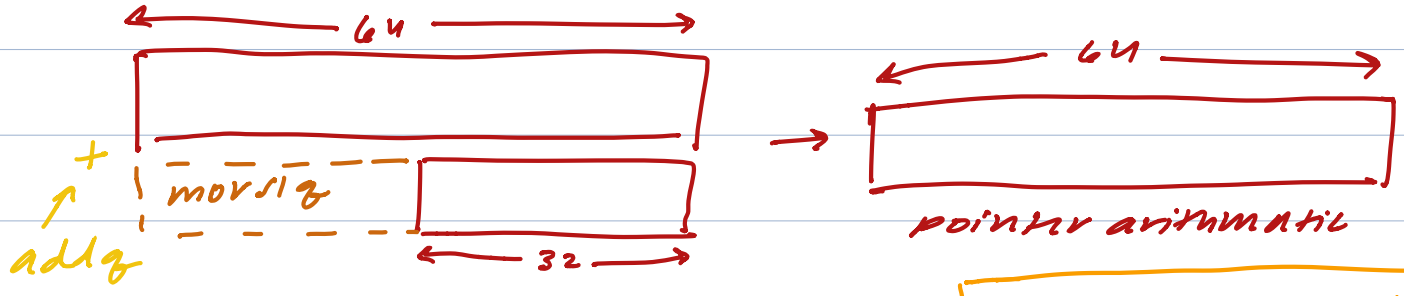
		src 2	src1 /dest
· Data manipulation	addl	a(%rip), %eax	

instruction pointer

· Control Flow	ret	returns

EX program #1

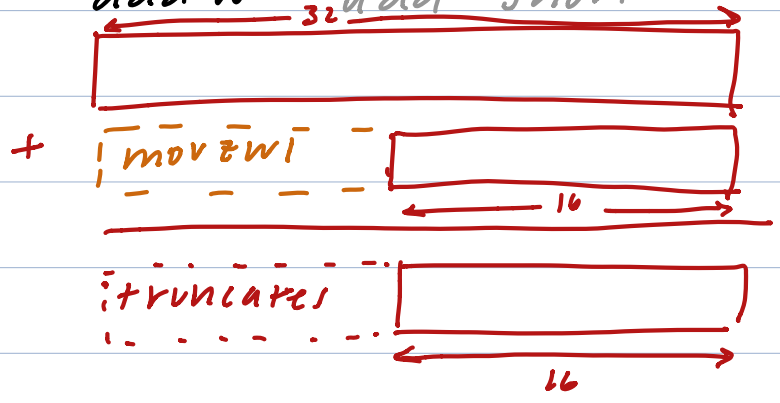
- movslq : move a signed long to qvad (rax)
- addq : add qvad (64 bit number)



EX program #2

- movzwl : move an unsigned w to l
- addw : add short

long	lld	32 bits
short	w	16 bits
qvad	q	64 bits
int	b	8 bits
	s	signed
	z	unsigned



New Program (more registers)

EX 3:

```
movl    %edi, %eax
ret
```

calling convention

eax return reg

edi 1st parameter reg

EX 4:

```
movl    %edi, -4(%rsp)
movl    %esi, -8(%rsp)
movl    -4(%rsp), %edx
movl    -8(%rsp), %edx
addl    %edx, %eax
ret
```

%rsp ← register stack pointer

%esi ← second parameter register

EX 5:

effective address calculations

```
.LFB0:
movl    %edi, -4(%rsp)
movl    %esi, -8(%rsp)
movl    -4(%rsp), %edx
movl    -8(%rsp), %eax
addl    %edx, %eax
ret

.LFBD:
leal    (%rdi,%rsi), %eax
ret
```

same program

lea ≡ load effective address

- figure out when you would have gone to memory

if you did

default 0 if not there

default 1
can be 1, 2, 4, or 8

- offset(base-reg, index-reg, scale)
- effective address = base-reg + index-reg * scale + offset

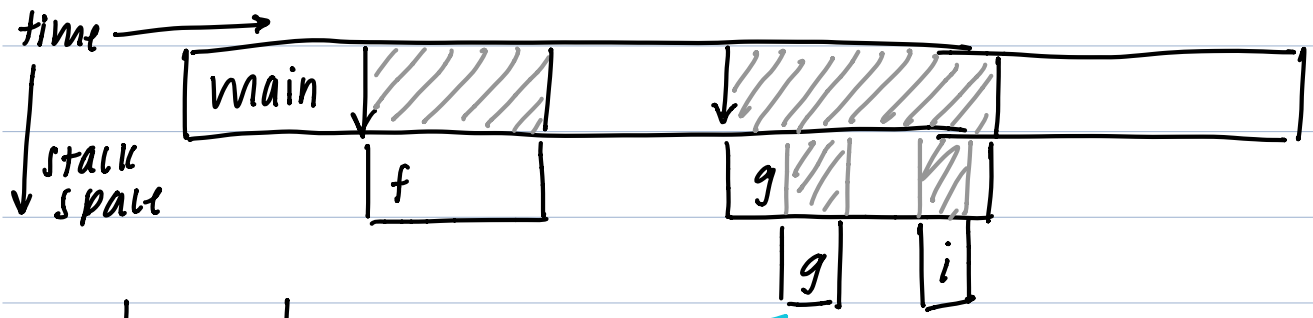
EX6:

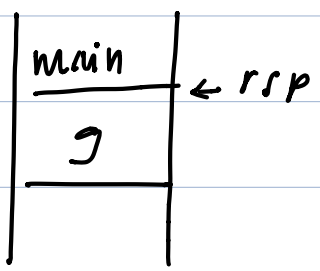
```
movl    %edi    a(%rip)
movl    %esi    4+a(%rip)
movl    %edx    8+a(%rip)
movl    %ecx    12+a(%rip)
movl    %r8d    16+a(%rip)
movl    %r9d    20+a(%rip)
ret
```

program:

```
a[0] = arg1;
a[1] = arg2;
   ⋮
a[6] = arg7;
```

stack





recursive call of g

stack (for ex 5)