Welcome to 213 recitation!

15/18-213: Introduction to Computer Systems
11\textsuperscript{st} Recitation, Sept. 26, 2011

\textbf{TA:} Pablo Chavez (pchavez)
Section E, 1:30pm – 2:20pm GHC 5222

Slide material from Kevin Su
Outline

• Reminders
• Stack discipline (x86)
• Stack discipline (x86_64)
• Data lab handback
Reminder – Bomb lab

• Due tomorrow, at midnight

• Feel free to ping the staff list with questions
  • 15-213-staff@cs.cmu.edu

• Office hours
  • Sun – Thurs 5:30pm – 8:30pm @ WeH 5207
Stacks
Stacks

Reminders  Stack x86  Stack x86_64  Data lab  Questions
Stacks (x86)

ebp->
esp->

Reminders  Stack x86  Stack x86_64  Data lab  Questions
Stack Operations (x86) – Push

earlier frames (including main)

ebp ->
stuff

esp ->

foo:

...  
movl $0x5, %edx
push %edx

Questions
Stack Operations (x86) – Push (2)

earlier frames (including main)

stuff

foo:

... 
movl $0x5, %edx
push %edx

Carnegie Mellon

foo frame
Stack Operations (x86) – Push (3)

earlier frames (including main)

ebp->

stuff

esp->

earlier frames (including main)

ebp->

stuff

esp->

foo:

...  
movl $0x5, %edx  
push %edx

foo frame
Stack Operations (x86) – Push (4)

What instructions are equivalent to `push %edx`?

```
foo:
  ...
  movl $0x5, %edx
  push %edx
```

Earlier frames (including main)

- **ebp** ->
- **esp** ->
  - `stuff`

Earlier frames (including main)

- **ebp** ->
- **esp** ->
  - `5`
Stack Operations (x86) – Pop
Stack Operations (x86) – Pop

Run previous slides backwards
Stack (x86) – High Level

Disclaimer: Next slides are for your conceptual understanding only. They are not representative of anything about order of execution.
Stack (x86) – High Level

```c
int add3(int a, int b, int c)
{
    return a + b + c;
}

int foo(int x, int y)
{
    int b = 5;
    int s = add3(x, y, b);
    return x >> s;
}
```

Reminders

- Stack x86
- Stack x86_64
- Data lab
- Questions
int add3(int a, int b, int c)
{
    return a + b + c;
}

int foo(int x, int y)
{
    int b = 5;
    int s = add3(x, y, b);
    return x >> s;
}
Stack (x86) – High Level

```c
int add3(int a, int b, int c)
{
    return a + b + c;
}

int foo(int x, int y)
{
    int b = 5;
    int s = add3(x, y, b);
    return x >> s;
}
```
Stack (x86) – Zoomed in

add3:
    pushl %ebp
    movl %esp, %ebp
    movl 12(%ebp), %edx
    movl 8(%ebp), %eax
    addl %edx, %eax
    addl 16(%ebp), %eax
    popl %ebp
    ret

foo:
    pushl %ebp
    movl %esp, %ebp
    subl $12, %esp
    movl $5, 8(%esp)
    movl $4, 4(%esp)
    movl $3, (%esp)
    call add3
    movl %eax, sum
    movl sum, %eax
    leave
    ret

int sum;
int add3(int a, int b, int c)
{
    return a + b + c;
}

int foo(void)
{
    sum = add3(3, 4, 5);
    return sum;
}

example stolen from princeton lecture slides
Stack (x86) – Zoomed in

add3:
  pushl %ebp
  movl %esp, %ebp
  movl 12(%ebp), %edx
  movl 8(%ebp), %eax
  addl %edx, %eax
  addl 16(%ebp), %eax
  popl %ebp
  ret

foo:
  pushl %ebp
  movl %esp, %ebp
  subl $12, %esp
  movl $5, 8(%esp)
  movl $4, 4(%esp)
  movl $3, (%esp)
  call add3
  movl %eax, sum
  movl sum, %eax
  leave
  ret
Stack (x86) – Zoomed in

add3:
pushl %ebp
movl %esp, %ebp
movl 12(%ebp), %edx
movl 8(%ebp), %eax
addl %edx, %eax
addl 16(%ebp), %eax
popl %ebp
ret

foo:
pushl %ebp
movl %esp, %ebp
subl $12, %esp
movl $5, 8(%esp)
movl $4, 4(%esp)
movl $3, (%esp)
call add3
call add3
movl %eax, sum
movl sum, %eax
leave
ret
Stack (x86) – Zoomed in

add3:
pushl %ebp
movl %esp, %ebp
movl 12(%ebp), %edx
movl 8(%ebp), %eax
addl %edx, %eax
addl 16(%ebp), %eax
popl %ebp
ret

foo:
pushl %ebp
movl %esp, %ebp
subl $12, %esp
movl $5, 8(%esp)
movl $4, 4(%esp)
movl $3, (%esp)
call add3
movl %eax, sum
movl sum, %eax
leave
ret

Reminders Stack x86 Stack x86_64 Data lab Questions
Stack (x86) – Zoomed in

add3:
   pushl %ebp
   movl %esp, %ebp
   movl 12(%ebp), %edx
   movl 8(%ebp), %eax
   addl %edx, %eax
   addl 16(%ebp), %eax
   popl %ebp
   ret

foo:
   pushl %ebp
   movl %esp, %ebp
   subl $12, %esp
   movl $5, 8(%esp)
   movl $4, 4(%esp)
   movl $3, (%esp)
   call add3
   movl %eax, sum
   movl sum, %eax
   leave
   ret

Reminders   Stack x86   Stack x86_64   Data lab   Questions
Stack (x86) – Zoomed in

add3:
```assembly
pushl %ebp
movl %esp, %ebp
movl 12(%ebp), %edx
movl 8(%ebp), %eax
addl %edx, %eax
addl 16(%ebp), %eax
popl %ebp
ret
```

foo:
```assembly
pushl %ebp
movl %esp, %ebp
subl $12, %esp
movl $5, 8(%esp)
movl $4, 4(%esp)
movl $3, (%esp)
call add3
movl %eax, sum
movl sum, %eax
leave
ret
```
Stack (x86) – Zoomed in

```assembly
add3:
pushl %ebp
movl %esp, %ebp
movl 12(%ebp), %edx
movl 8(%ebp), %eax
addl %edx, %eax
addl 16(%ebp), %eax
popl %ebp
ret

foo:
pushl %ebp
movl %esp, %ebp
subl $12, %esp
movl $5, 8(%esp)
movl $4, 4(%esp)
movl $3, (%esp)
call add3
movl %eax, sum
movl sum, %eax
leave
ret
```

### Reminders

- Stack x86
- Stack x86_64
- Data lab
- Questions
Stack (x86) – Zoomed in

add3:
  pushl %ebp
  movl %esp, %ebp
  movl 12(%ebp), %edx
  movl 8(%ebp), %eax
  addl %edx, %eax
  addl 16(%ebp), %eax
  popl %ebp
  ret

foo:
  pushl %ebp
  movl %esp, %ebp
  subl $12, %esp
  movl $5, 8(%esp)
  movl $4, 4(%esp)
  movl $3, (%esp)
  call add3
  movl %eax, sum
  movl sum, %eax
  leave
  ret

Reminders
Stack x86
Stack x86_64
Data lab
Questions
Stack (x86) – Zoomed in

add3:
pushl %ebp
movl %esp, %ebp
movl 12(%ebp), %edx
movl 8(%ebp), %eax
addl %edx, %eax
addl 16(%ebp), %eax
popl %ebp
ret

foo:
pushl %ebp
movl %esp, %ebp
subl $12, %esp
movl $5, 8(%esp)
movl $4, 4(%esp)
movl $3, (%esp)
call add3
movl %eax, sum
movl sum, %eax
leave
ret

What happens here?

Reminders  Stack x86  Stack x86_64  Data lab  Questions
Stack (x86) – Zoomed in

add3:
  pushl %ebp
  movl %esp, %ebp
  movl 12(%ebp), %edx
  movl 8(%ebp), %eax
  addl %edx, %eax
  addl 16(%ebp), %eax
  popl %ebp
  ret

foo:
  pushl %ebp
  movl %esp, %ebp
  subl $12, %esp
  movl $5, 8(%esp)
  movl $4, 4(%esp)
  movl $3, (%esp)
  call add3
  <foo+ff> movl %eax, sum
  movl sum, %eax
  leave
  ret`
Stack (x86) – Zoomed in

add3:
    pushl %ebp
    movl %esp, %ebp
    movl 12(%ebp), %edx
    movl 8(%ebp), %eax
    addl %edx, %eax
    addl 16(%ebp), %eax
    popl %ebp
    ret

foo:
    pushl %ebp
    movl %esp, %ebp
    subl $12, %esp
    movl $5, 8(%esp)
    movl $4, 4(%esp)
    movl $3, (%esp)
    call add3
    <foo+ff> movl %eax, sum
    movl sum, %eax
    leave
    ret`

What two instructions was that equivalent to?
Stack (x86) – Zoomed in

add3:
  pushl %ebp ≤
  movl %esp, %ebp
  movl 12(%ebp), %edx
  movl 8(%ebp), %eax
  addl %edx, %eax
  addl 16(%ebp), %eax
  popl %ebp
  ret

foo:
  pushl %ebp
  movl %esp, %ebp
  subl $12, %esp
  movl $5, 8(%esp)
  movl $4, 4(%esp)
  movl $3, (%esp)
  call add3
  <foo+ff> movl %eax, sum
  movl sum, %eax
  leave
  ret`

Reminders

Stack x86

Stack x86_64

Data lab

Questions
Stack (x86) – Zoomed in

add3:
  pushl %ebp
  movl %esp, %ebp
  movl 12(%ebp), %edx
  movl 8(%ebp), %eax
  addl %edx, %eax
  addl 16(%ebp), %eax
  popl %ebp
  ret

foo:
  pushl %ebp
  movl %esp, %ebp
  subl $12, %esp
  movl $5, 8(%esp)
  movl $4, 4(%esp)
  movl $3, (%esp)
  call add3
  <foo+ff> movl %eax, sum
  movl sum, %eax
  leave
  ret`
Stack (x86) – Zoomed in

add3:
  pushl %ebp
  movl %esp, %ebp
  movl 12(%ebp), %edx
  movl 8(%ebp), %eax
  addl %edx, %eax
  addl 16(%ebp), %eax
  popl %ebp
  ret

foo:
  pushl %ebp
  movl %esp, %ebp
  subl $12, %esp
  movl $5, 8(%esp)
  movl $4, 4(%esp)
  movl $3, (%esp)
  call add3
  <foo+ff> movl %eax, sum
  movl sum, %eax
  leave
  ret

Reminders
Stack x86
Stack x86_64
Data lab
Questions
Stack (x86) – Zoomed in

add3:
    pushl %ebp
    movl %esp, %ebp
    movl 12(%ebp), %edx
    movl 8(%ebp), %eax
    addl %edx, %eax
    addl 16(%ebp), %eax
    popl %ebp
    ret

foo:
    pushl %ebp
    movl %esp, %ebp
    subl $12, %esp
    movl $5, 8(%esp)
    movl $4, 4(%esp)
    movl $3, (%esp)
    call add3
<foo+ff> movl %eax, sum
    movl sum, %eax
    leave
    ret`
Stack (x86) – Zoomed in

add3:
  pushl %ebp
  movl %esp, %ebp
  movl 12(%ebp), %edx
  movl 8(%ebp), %eax
  addl %edx, %eax
  addl 16(%ebp), %eax
  popl %ebp
  ret

foo:
  pushl %ebp
  movl %esp, %ebp
  subl $12, %esp
  movl $5, 8(%esp)
  movl $4, 4(%esp)
  movl $3, (%esp)
  call add3
  <foo+ff> movl %eax, sum
  movl sum, %eax
  leave
  ret`
Stack (x86) – Zoomed in

add3:

```assembly
pushl %ebp
movl %esp, %ebp
movl 12(%ebp), %edx
movl 8(%ebp), %eax
addl %edx, %eax
addl 16(%ebp), %eax
popl %ebp
ret
```

foo:

```assembly
pushl %ebp
movl %esp, %ebp
subl $12, %esp
movl $5, 8(%esp)
movl $4, 4(%esp)
movl $3, (%esp)
call add3
<foo+ff> movl %eax, sum
movl sum, %eax
leave
ret`
```

What does ret need to do?

---

Reminders  Stack x86  Stack x86_64  Data lab  Questions
Stack (x86) – Zoomed in

add3:
  pushl %ebp
  movl %esp, %ebp
  movl 12(%ebp), %edx
  movl 8(%ebp), %eax
  addl %edx, %eax
  addl 16(%ebp), %eax
  popl %ebp
  ret

foo:
  pushl %ebp
  movl %esp, %ebp
  subl $12, %esp
  movl $5, 8(%esp)
  movl $4, 4(%esp)
  movl $3, (%esp)
  call add3
  <foo+ff> movl %eax, sum ←
  movl sum, %eax
  leave
  ret

Reminders Stack x86 Stack x86_64 Data lab Questions
Stack (x86) – Zoomed in

add3:
  pushl %ebp
  movl %esp, %ebp
  movl 12(%ebp), %edx
  movl 8(%ebp), %eax
  addl %edx, %eax
  addl 16(%ebp), %eax
  popl %ebp
  ret

foo:
  pushl %ebp
  movl %esp, %ebp
  subl $12, %esp
  movl $5, 8(%esp)
  movl $4, 4(%esp)
  movl $3, (%esp)
  call add3
  movl %eax, sum
  movl sum, %eax
  leave
  ret`

Reminders  Stack x86  Stack x86_64  Data lab  Questions
Stack (x86) – Zoomed in

add3:

```assembly
pushl %ebp
movl %esp, %ebp
movl 12(%ebp), %edx
movl 8(%ebp), %eax
addl %edx, %eax
addl 16(%ebp), %eax
popl %ebp
ret
```

foo:

```assembly
pushl %ebp
movl %esp, %ebp
subl $12, %esp
movl $5, 8(%esp)
movl $4, 4(%esp)
movl $3, (%esp)
call add3
<foo+ff> movl %eax, sum
movl sum, %eax
leave
ret`
```

<table>
<thead>
<tr>
<th>main frame</th>
<th>0x79b4</th>
<th>0x7900</th>
<th>0x78fc</th>
</tr>
</thead>
<tbody>
<tr>
<td>ebp</td>
<td>0x79b4 (old ebp)</td>
<td>0x7900</td>
<td>0x78fc</td>
</tr>
<tr>
<td>esp</td>
<td>0x78f8</td>
<td>0x78f8</td>
<td>0x78fc</td>
</tr>
<tr>
<td>5 (arg 3)</td>
<td>0x78f0</td>
<td>0x78f0</td>
<td>0x78fc</td>
</tr>
<tr>
<td>4 (arg 2)</td>
<td>0x78f0</td>
<td>0x78f0</td>
<td>0x78fc</td>
</tr>
<tr>
<td>3 (arg 1)</td>
<td>0x78f0</td>
<td>0x78f0</td>
<td>0x78fc</td>
</tr>
<tr>
<td>&lt;foo + ff&gt; (ret addr)</td>
<td>0x78ec</td>
<td>0x78ec</td>
<td>0x78fc</td>
</tr>
<tr>
<td>0x78fc (old ebp)</td>
<td>0x7888</td>
<td>0x7888</td>
<td>0x78fc</td>
</tr>
</tbody>
</table>

Reminders Stack x86 Stack x86_64 Data lab Questions
Stack (x86) – Zoomed in

add3:
```
pushl %ebp
movl %esp, %ebp
movl 12(%ebp), %edx
movl 8(%ebp), %eax
addl %edx, %eax
addl 16(%ebp), %eax
popl %ebp
ret
```

foo:
```
pushl %ebp
movl %esp, %ebp
subl $12, %esp
movl $5, 8(%esp)
movl $4, 4(%esp)
movl $3, (%esp)
call add3
<foo+ff> movl %eax, sum
movl sum, %eax
leave
ret`
```
Stack (x86) – Zoomed in

```
add3:
pushl %ebp
movl %esp, %ebp
movl 12(%ebp), %edx
movl 8(%ebp), %eax
addl %edx, %eax
addl 16(%ebp), %eax
popl %ebp
ret

foo:
pushl %ebp
movl %esp, %ebp
subl $12, %esp
movl $5, 8(%esp)
movl $4, 4(%esp)
movl $3, (%esp)
call add3
<foo + ff> movl %eax, sum
movl sum, %eax
leave
ret`
```

<table>
<thead>
<tr>
<th>ebp -&gt;</th>
<th>esp -&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x79b4</td>
<td>0x78f8</td>
</tr>
<tr>
<td>0x7900</td>
<td>0x78f4</td>
</tr>
<tr>
<td>0x79b4 (old ebp)</td>
<td>0x78f0</td>
</tr>
<tr>
<td>5 (arg 3)</td>
<td>0x78f0</td>
</tr>
<tr>
<td>4 (arg 2)</td>
<td>0x78f0</td>
</tr>
<tr>
<td>3 (arg 1)</td>
<td>0x78f0</td>
</tr>
<tr>
<td>&lt;foo + ff&gt; (ret addr)</td>
<td>0x78f0</td>
</tr>
<tr>
<td>0x78fc (old ebp)</td>
<td>0x78f0</td>
</tr>
</tbody>
</table>
Stack (x86) – Zoomed in

We’re about to (finally) move on. Any questions about that diagram?
Stack (x86) – Local variables

add3:
    pushl %ebp
    movl %esp, %ebp
    movl 12(%ebp), %edx
    movl 8(%ebp), %eax
    addl %edx, %eax
    addl 16(%ebp), %eax
    popl %ebp
    ret

foo:
    pushl %ebp
    movl %esp, %ebp
    subl $12, %esp
    movl $5, 8(%esp)
    movl $4, 4(%esp)
    movl $3, (%esp)
    call add3
    <foo+ff> movl %eax, sum
    movl sum, %eax
    leave
    ret`

int sum;
int add3(int a, int b, int c)
{
    return a + b + c;
}

int foo(void)
{
    sum = add3(3, 4, 5);
    return sum;
}

example stolen from princeton lecture slides
Stack (x86) – Local variables

```c
int sum;
int add3(int a, int b, int c)
{
    return a + b + c;
}

int foo(void)
{
    int sum;
    sum = add3(3, 4, 5);
    return sum;
}
```

Example stolen from Princeton lecture slides
Stack (x86) – Local variables

Show: unsafe.c
Stack (x86) – Registers

- eax
  - return value

- eax, ecx, edx
  - caller save

- ebx, edi, esi
  - callee save
Stack (x86_x64) – Registers

- A simplification of x86
- rax
  - return value
- rax, r10, r11
  - caller save
- rbx, rbp, r12 – r15
  - callee save
- rdi – arg1
- rsi – arg2
- rdx – arg3
- rcx – arg4
- r8 – arg5
- r9 – arg6
Stack (x86_x64) – Stack Allocation

- Often none.
Stack (x86_x64) – Stack Allocation

- Often none.

- No use of %rbp
Stack (x86_x64) – Stack Allocation

- Often none.
- No use of %rbp
Stack (x86_x64) – Stack Allocation

- Often none.
- No use of %rbp
- Say need $5, $4, $3 on stack: rsp ->

other frames
Stack (x86_x64) – Stack Allocation

- Often none.
- No use of %rbp
- Say need $5, $4, $3 on stack: 
  \[
  \text{rsp} \rightarrow \]
  
  \(5\)

other frames
Stack (x86_x64) – Stack Allocation

• Often none.
• No use of %rbp
• Say need $5, $4, $3 on stack:

other frames

5

rsp ->
Stack (x86_x64) – Stack Allocation

- Often none.
- No use of %rbp
- Say need $5, $4, $3 on stack:

```
rsp ->
  5
  4
```

other frames
Stack (x86_x64) – Stack Allocation

- Often none.
- No use of %rbp

Say need $5, $4, $3 on stack:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th>other frames</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rsp-&gt;</td>
<td></td>
<td></td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Reminders  Stack x86  Stack x86_64  Data lab  Questions
Stack (x86_x64) – Stack Allocation

- Often none.
- No use of %rbp
- When done?

\[ \begin{array}{c}
\text{other frames} \\
5 \\
4 \\
3 \\
\end{array} \]

\[ \begin{array}{c}
\text{rsp} \rightarrow 
\end{array} \]
Stack (x86_x64) – Stack Allocation

- Often none.
- No use of %rbp
- When done:

```
 rsp ->

  5

  4

  3

other frames
```
Data lab handback

Common things:
wrap lines to 80 characters!

/* calculate result */
return ~((x >> 31) | (!!x << 31)) >> n
+-0) & 0x841fffff;

if(x)
{
    return 0;
}
else
{
    statement1;
    statement2;
    ...
    statement52;
}
Questions?