15213 Recitation Section C

Shimin Chen

Sept. 23, 2002

Outline

- Last week's exercise
- Function and stack
- Array
- Struct and linked-list

Last Week's Final Example

```
int func5(int x){ ??? }
```

```
0x80483c3   mov   0x8(%ebp),%ecx
0x80483c6   xor   %eax,%eax
0x80483c8   xor   %edx,%edx
0x80483ca   cmp   %ecx,%edx
0x80483cc   jge   0x80483d7
0x80483ce   mov   %esi,%esi
0x80483d0   add   %edx,%eax
0x80483d2   inc   %edx
0x80483d3   cmp   %ecx,%edx
0x80483d5   jl    0x80483d0
```

Body

Write Comments

```
int func5(int x){ ??? }
```

```
0x80483c3   mov   0x8(%ebp),%ecx   ecx = x
0x80483c6   xor   %eax,%eax        eax = 0
0x80483c8   xor   %edx,%edx        edx = 0
0x80483ca   cmp   %ecx,%edx        if (edx>=x)
0x80483cc   jge   0x80483d7          goto L1
0x80483ce   mov   %esi,%esi        nop
0x80483d0   add   %edx,%eax        L2:eax += edx
0x80483d2   inc   %edx             edx ++
0x80483d3   cmp   %ecx,%edx        if (edx<x)
0x80483d5   jl    0x80483d0          goto L2
0x80483d7   ---                    L1:
```


Name the variables

- eax– result, edx--i

```
0x80483c3   mov   0x8(%ebp),%ecx   ecx = x;
0x80483c6   xor   %eax,%eax        result = 0;
0x80483c8   xor   %edx,%edx        i = 0;
0x80483ca   cmp   %ecx,%edx        if (i>=x)
0x80483cc   jge   0x80483d7          goto L1;
0x80483ce   mov   %esi,%esi
0x80483d0   add   %edx,%eax        L2:result += i;
0x80483d2   inc   %edx               i++;
0x80483d3   cmp   %ecx,%edx        if (i<x)
0x80483d5   jl    0x80483d0          goto L2;
0x80483d7   ---                    L1:
```

Loop

```
result = 0;
i = 0;
if (i>=x)
  goto L1;

L2:result += i;
   i++;
if (i<x)
  goto L2;
L1:
```

```
result = 0; i = 0;
if (i>=x)  goto L1;
do {
     result += i;
     i++;
}while (i<x);
L1:
```

```
result = 0; i = 0;
While (i<x){
     result += i;
     i++;
}
```

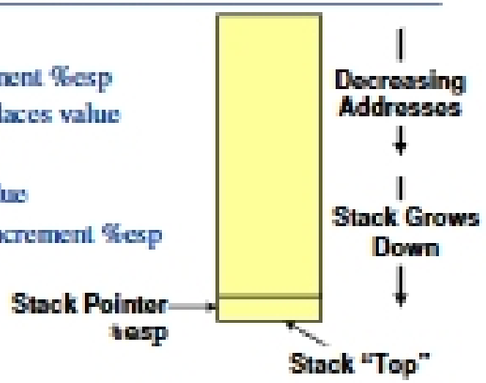
```
result = 0;
for (i=0; i<x; i++)
     result += i;
```


C Code

```
int func5(int x)
{
     int result=0;
     int i;
     for (i=0; i<x; i++)
        result += i;
     return result;
}
```

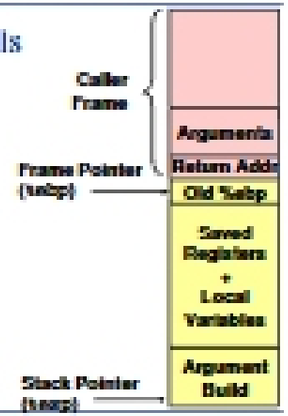

Stack Basics

- push
  - decrement %esp
  - then places value
- pop
  - get value
  - then increment %esp



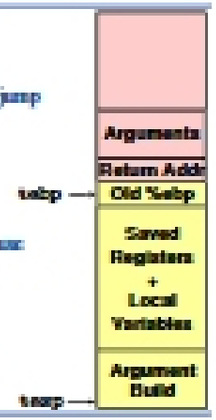
Function Stack Frames

- A caller function calls a callee function



Making a Call

- Caller:
  - "push" arguments (*in what order?*)
  - "call": put *return address* onto stack, jump to the start of callee function
- Callee:
  - save (caller's) %ebp
  - set up stack frame
  - save *callee-saved* registers if want to use:
    - %ebx, %esi, %edi
  - put return value in %eax
  - restore %ebp and %esp
  - "ret" to jump to the "Return Addr"



Example 1

- Please draw the stack at the marked points
- Write C code for the assembly code
  - (gdb) x/s 0x8048478
  - 0x8048478 <_IO_stdin_used+4>: "%d\n"

```
int example_1 (int x, int y)
0x080483e4 <example_1>:    push   %ebp
0x080483e5 <example_1+1>:  mov    %esp,%ebp
0x080483e7 <example_1+3>:  mov    0xc(%ebp),%eax
0x080483ea <example_1+6>:  add    0x8(%ebp),%eax
0x080483ed <example_1+9>:  mov    %ebp,%esp
0x080483ef <example_1+11>: pop    %ebp
0x080483f0 <example_1+12>: ret
```

2.Stack?

ASM of main()

```
0x080483f4 <main>:         push   %ebp
0x080483f5 <main+1>:       mov    %esp,%ebp
0x080483f7 <main+3>:       sub    $0x8,%esp
0x080483fa <main+6>:       add    $0xfffffff8,%esp
0x080483fd <main+9>:       push   $0x2
0x080483ff <main+11>:      push   $0x1
0x08048401 <main+13>:      call   0x80483e4 <example_1>
0x08048406 <main+18>:      add    $0xfffffff8,%esp
0x08048409 <main+21>:      push   %eax
0x0804840a <main+22>:      push   $0x8048478
0x0804840f <main+27>:      call   0x8048300 <printf>
0x08048414 <main+32>:      xor    %eax,%eax
0x08048416 <main+34>:      mov    %ebp,%esp
0x08048418 <main+36>:      pop    %ebp
0x08048419 <main+37>:      ret
```

1.Stack?
3.Stack?

Stack at Point 1

```
<main>:
0x80e83f4:  push   %ebp
0x80e83f5:  mov    %esp,%ebp
0x80e83f7:  sub    $0x8,%esp
0x80e83fa:  add    $0xfffffff8,%esp
0x80e83fd:  push   $0x2
0x80e83ff:  push   $0x1
0x80e8401:  call   0x80e83e4
  <example_1>
0x80e8406:  ——
```