MP1

<checkpoint 1>

CS461 / ECE422 – UIUC Spring 2016

By: Gene Shiue

Outline

-GDB

-Stack frame + x86 assembly

-Endianness

-Shellcode

GDB?

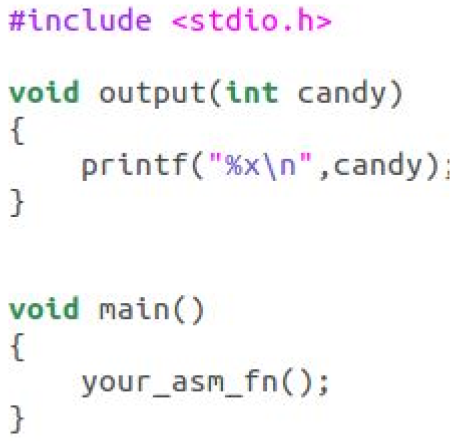
- Debugger
- Stop/pause programs
- Examine memory / registers

- Find bugs!
- Context of MP: find vulnerabilities

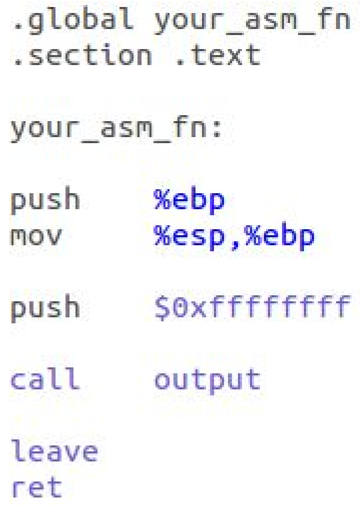
GDB Tutorial - Important Commands

- Disassemble: *disas function_name*
- Set breakpoints: *b function_name*, *b *0xbffebee0*
- Examine: *x $eax, x/s $esp, x/wx 0xdeadbeef, x/2wx 0x5adface5*
- Look at register values: *info reg*
- Run: *r*
- Continue: *c*
- Step(one instruction): *si*
- Show current instruction: *display/i $pc*

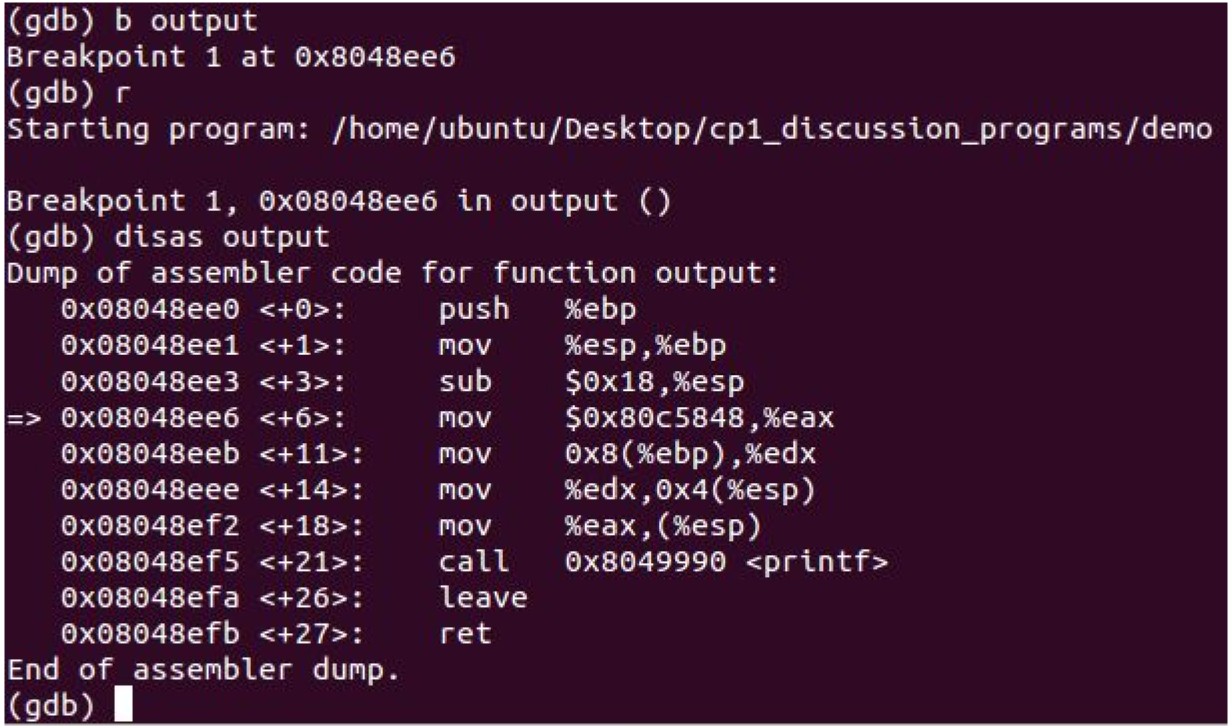
Exercise



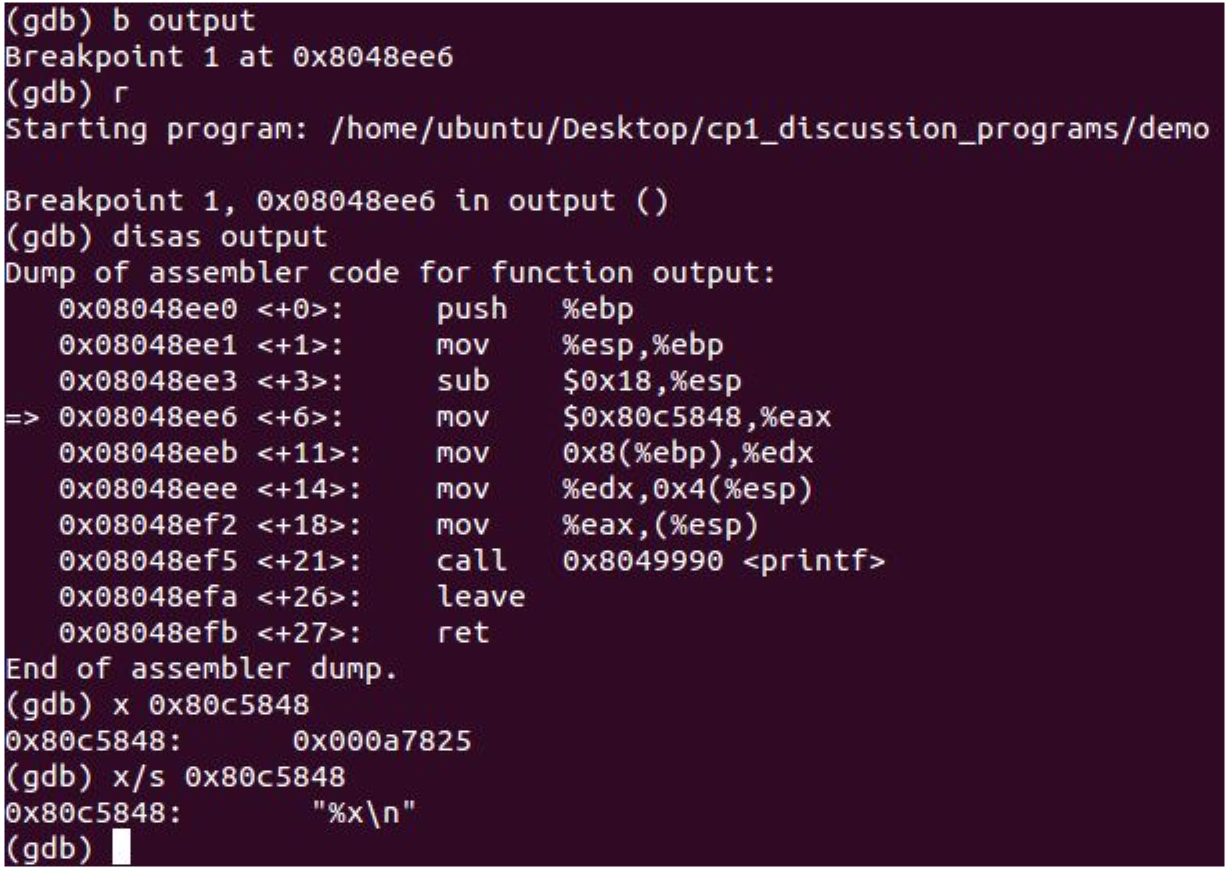
```c
#include <stdio.h>

void output(int candy)
{
    printf("%x\n",candy);
}


void main()
{
    your_asm_fn();
}
```



```asm
.global your_asm_fn
.section .text

your_asm_fn:

push    %ebp
mov     %esp,%ebp

push    $0xffffffff

call    output

leave
ret
```



```
(gdb) b output
Breakpoint 1 at 0x8048ee6
(gdb) r
Starting program: /home/ubuntu/Desktop/cp1_discussion_programs/demo

Breakpoint 1, 0x08048ee6 in output ()
(gdb) disas output
Dump of assembler code for function output:
   0x08048ee0 <+0>:     push   %ebp
   0x08048ee1 <+1>:     mov    %esp,%ebp
   0x08048ee3 <+3>:     sub    $0x18,%esp
=> 0x08048ee6 <+6>:     mov    $0x80c5848,%eax
   0x08048eeb <+11>:    mov    0x8(%ebp),%edx
   0x08048eee <+14>:    mov    %edx,0x4(%esp)
   0x08048ef2 <+18>:    mov    %eax,(%esp)
   0x08048ef5 <+21>:    call   0x8049990 <printf>
   0x08048efa <+26>:    leave
   0x08048efb <+27>:    ret
End of assembler dump.
(gdb) 
```



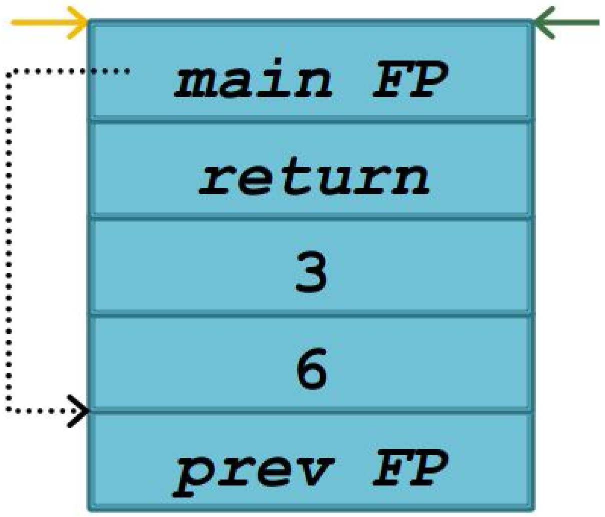
```
(gdb) b output
Breakpoint 1 at 0x8048ee6
(gdb) r
Starting program: /home/ubuntu/Desktop/cp1_discussion_programs/demo

Breakpoint 1, 0x08048ee6 in output ()
(gdb) disas output
Dump of assembler code for function output:
   0x08048ee0 <+0>:     push   %ebp
   0x08048ee1 <+1>:     mov    %esp,%ebp
   0x08048ee3 <+3>:     sub    $0x18,%esp
=> 0x08048ee6 <+6>:     mov    $0x80c5848,%eax
   0x08048eeb <+11>:    mov    0x8(%ebp),%edx
   0x08048eee <+14>:    mov    %edx,0x4(%esp)
   0x08048ef2 <+18>:    mov    %eax,(%esp)
   0x08048ef5 <+21>:    call   0x8049990 <printf>
   0x08048efa <+26>:    leave
   0x08048efb <+27>:    ret
End of assembler dump.
(gdb) x 0x80c5848
0x80c5848:      0x000a7825
(gdb) x/s 0x80c5848
0x80c5848:       "%x\n"
(gdb) 
```

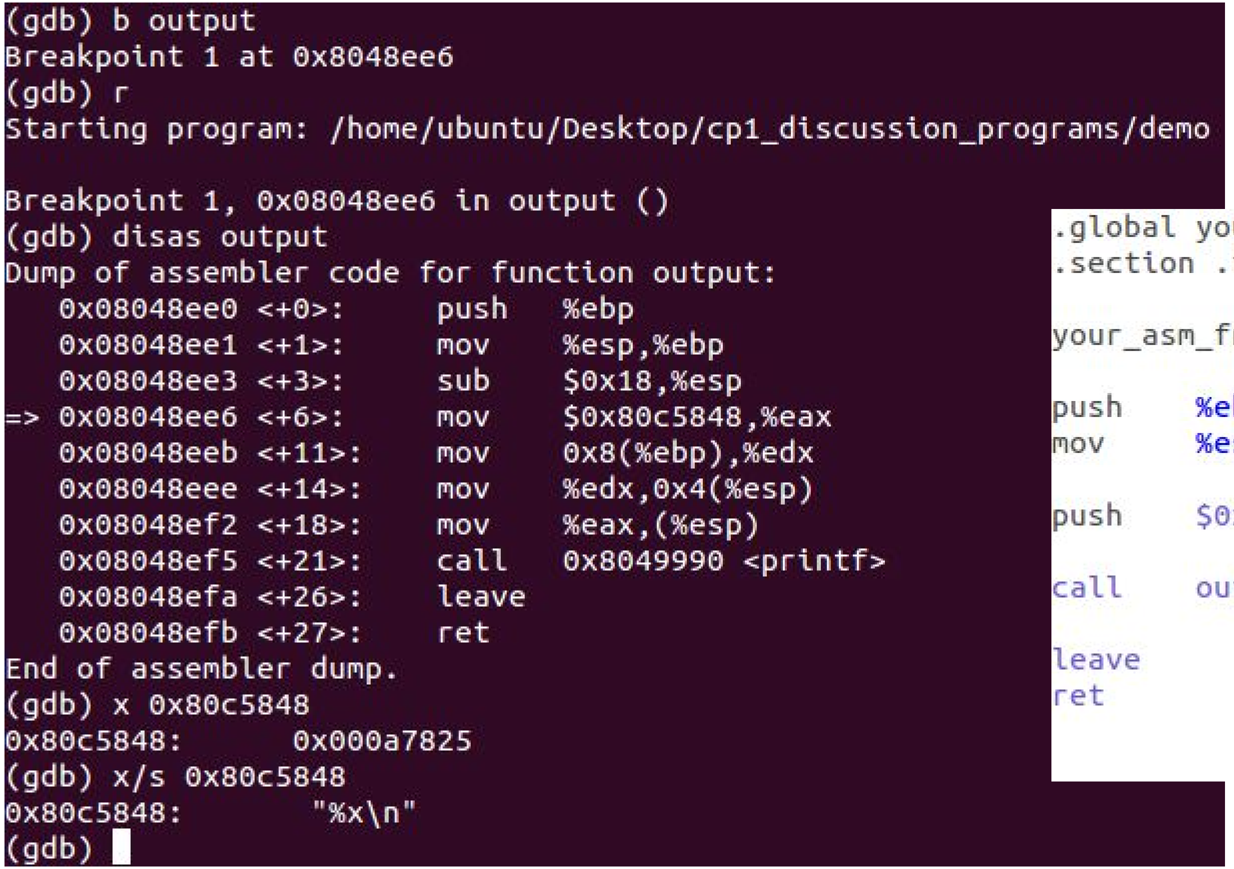
example.c

```c
void foo(int a, int b) {
    char buf1[10];
}

void main() {
    foo(3,6);
}
```

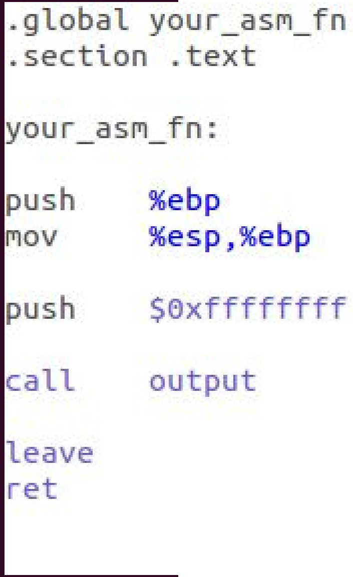
example.s (x86)

```
foo:
    pushl   %ebp
    movl    %esp, %ebp
    subl    $16, %esp
    leave
    ret
```

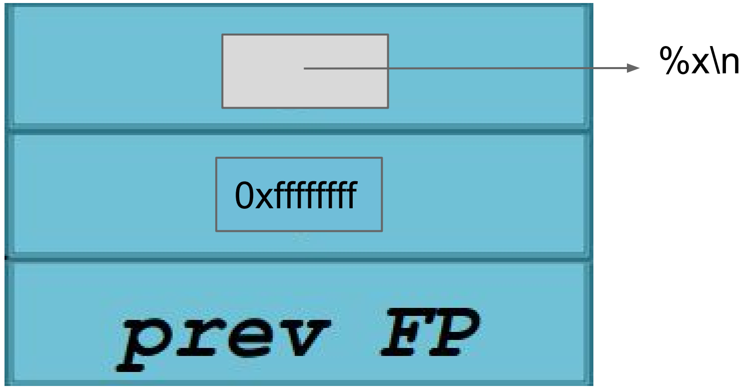


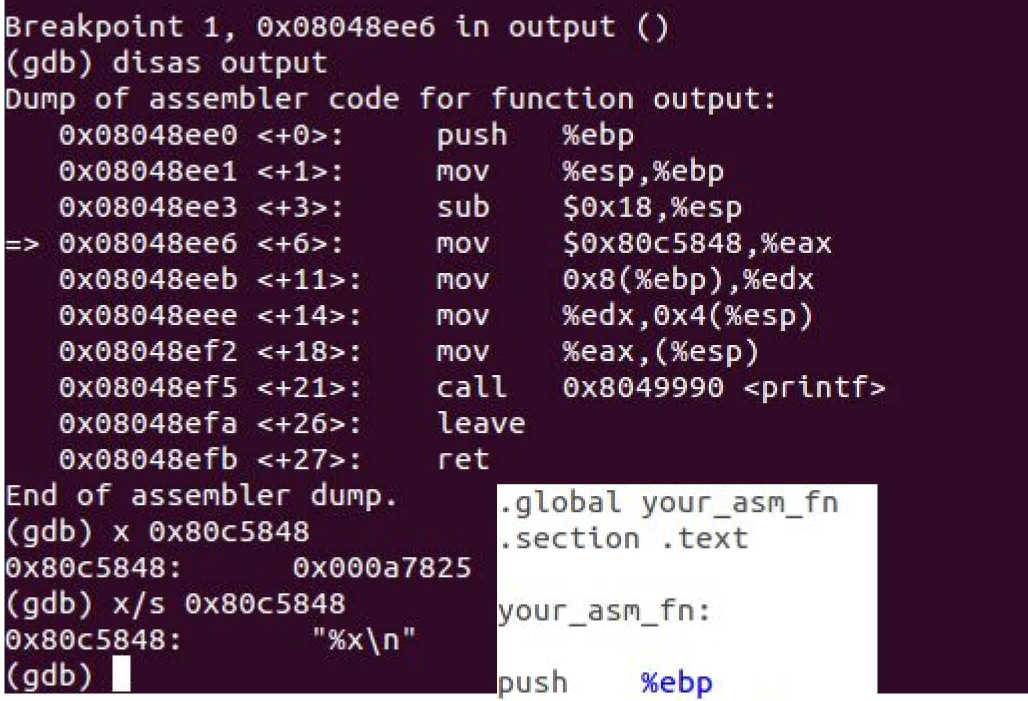


```
(gdb) b output
Breakpoint 1 at 0x8048ee6
(gdb) r
Starting program: /home/ubuntu/Desktop/cp1_discussion_programs/demo

Breakpoint 1, 0x08048ee6 in output ()
(gdb) disas output
Dump of assembler code for function output:
   0x08048ee0 <+0>:     push   %ebp
   0x08048ee1 <+1>:     mov    %esp,%ebp
   0x08048ee3 <+3>:     sub    $0x18,%esp
=> 0x08048ee6 <+6>:     mov    $0x80c5848,%eax
   0x08048eeb <+11>:    mov    0x8(%ebp),%edx
   0x08048eee <+14>:    mov    %edx,0x4(%esp)
   0x08048ef2 <+18>:    mov    %eax,(%esp)
   0x08048ef5 <+21>:    call   0x8049990 <printf>
   0x08048efa <+26>:    leave
   0x08048efb <+27>:    ret
End of assembler dump.
(gdb) x 0x80c5848
0x80c5848:      0x000a7825
(gdb) x/s 0x80c5848
0x80c5848:       "%x\n"
(gdb) 
```



```
.global your_asm_fn
.section .text

your_asm_fn:

push    %ebp
mov     %esp,%ebp

push    $0xffffffff

call    output

leave
ret
```





```
Breakpoint 1, 0x08048ee6 in output ()
(gdb) disas output
Dump of assembler code for function output:
   0x08048ee0 <+0>:     push   %ebp
   0x08048ee1 <+1>:     mov    %esp,%ebp
   0x08048ee3 <+3>:     sub    $0x18,%esp
=> 0x08048ee6 <+6>:     mov    $0x80c5848,%eax
   0x08048eeb <+11>:    mov    0x8(%ebp),%edx
   0x08048eee <+14>:    mov    %edx,0x4(%esp)
   0x08048ef2 <+18>:    mov    %eax,(%esp)
   0x08048ef5 <+21>:    call   0x8049990 <printf>
   0x08048efa <+26>:    leave
   0x08048efb <+27>:    ret
End of assembler dump.
(gdb) x 0x80c5848
0x80c5848:      0x000a7825
(gdb) x/s 0x80c5848
0x80c5848:       "%x\n"
(gdb)
```

```
.global your_asm_fn
.section .text

your_asm_fn:

push    %ebp
mov     %esp,%ebp

push    $0xffffffff

call    output

leave
ret
```

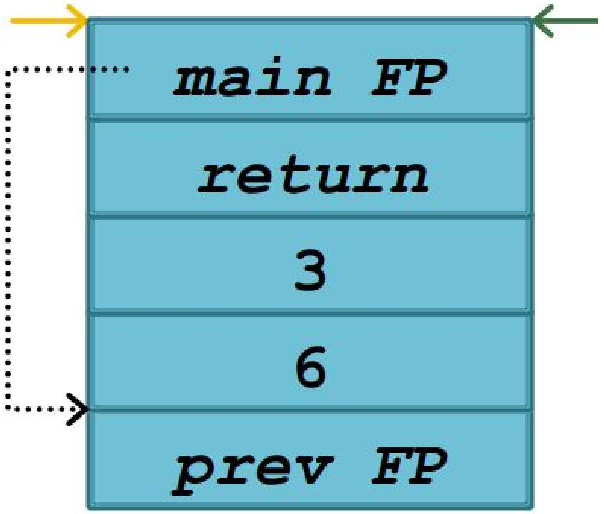
Exercise - ???

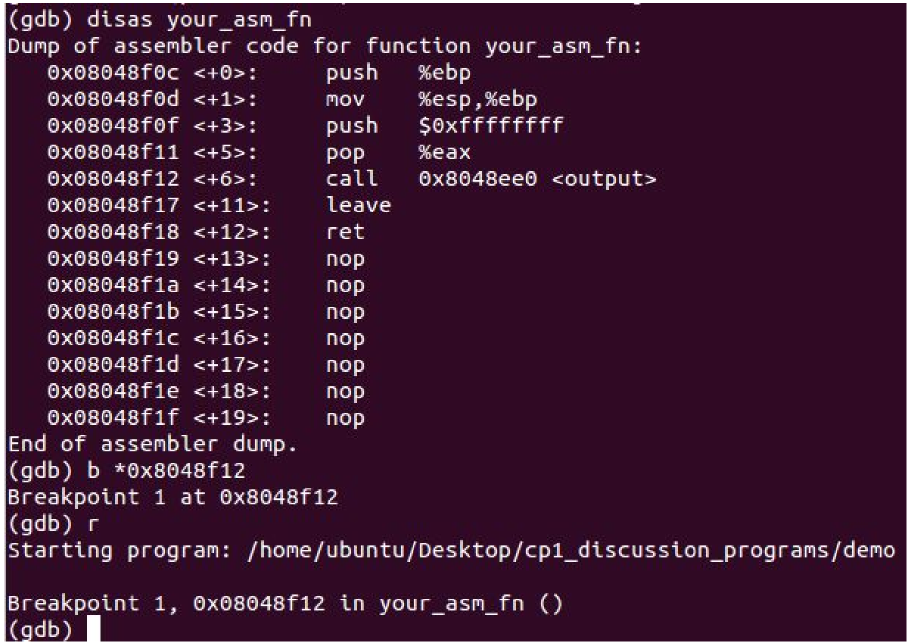
```c
#include <stdio.h>

void output(int candy)
{
    printf("%x\n",candy);
}


void main()
{
    your_asm_fn();
}
```

```
.global your_asm_fn
.section .text

your_asm_fn:

push    %ebp
mov     %esp,%ebp

push    $0xffffffff
pop     %eax

call    output

leave
ret
```

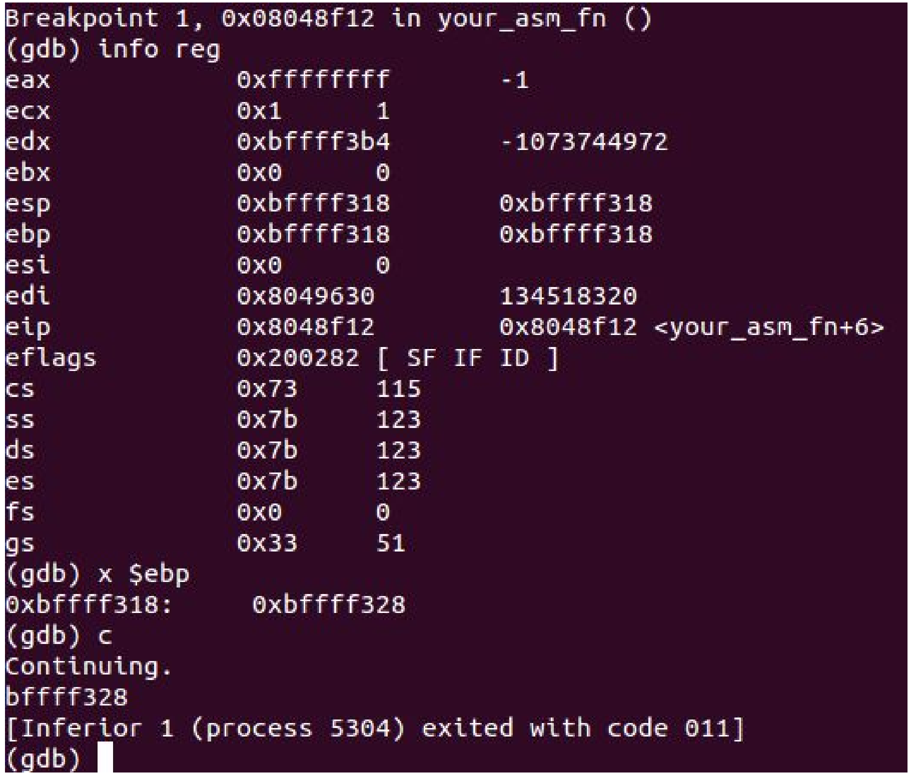
example.s (x86)

```
foo:
    pushl    %ebp
    movl     %esp, %ebp
    subl     $16, %esp
    leave
    ret
```

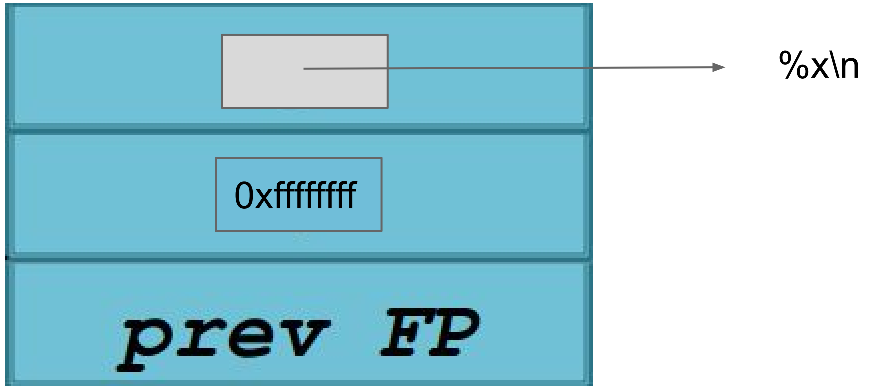




```
(gdb) disas your_asm_fn
Dump of assembler code for function your_asm_fn:
   0x08048f0c <+0>:     push   %ebp
   0x08048f0d <+1>:     mov    %esp,%ebp
   0x08048f0f <+3>:     push   $0xffffffff
   0x08048f11 <+5>:     pop    %eax
   0x08048f12 <+6>:     call   0x8048ee0 <output>
   0x08048f17 <+11>:    leave
   0x08048f18 <+12>:    ret
   0x08048f19 <+13>:    nop
   0x08048f1a <+14>:    nop
   0x08048f1b <+15>:    nop
   0x08048f1c <+16>:    nop
   0x08048f1d <+17>:    nop
   0x08048f1e <+18>:    nop
   0x08048f1f <+19>:    nop
End of assembler dump.
(gdb) b *0x8048f12
Breakpoint 1 at 0x8048f12
(gdb) r
Starting program: /home/ubuntu/Desktop/cp1_discussion_programs/demo

Breakpoint 1, 0x08048f12 in your_asm_fn ()
(gdb) 
```



```
Breakpoint 1, 0x08048f12 in your_asm_fn ()
(gdb) info reg
eax            0xffffffff       -1
ecx            0x1      1
edx            0xbffff3b4       -1073744972
ebx            0x0      0
esp            0xbffff318       0xbffff318
ebp            0xbffff318       0xbffff318
esi            0x0      0
edi            0x8049630        134518320
eip            0x8048f12        0x8048f12 <your_asm_fn+6>
eflags         0x200282 [ SF IF ID ]
cs             0x73     115
ss             0x7b     123
ds             0x7b     123
es             0x7b     123
fs             0x0      0
gs             0x33     51
(gdb) x $ebp
0xbffff318:     0xbffff328
(gdb) c
Continuing.
bffff328
[Inferior 1 (process 5304) exited with code 011]
(gdb) 
```

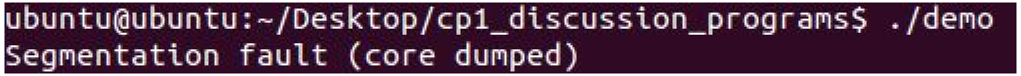

%x\n
0xffffffff
prev FP

```c
#include <stdio.h>

void output(int candy)
{
    printf("%x\n",candy);
}


void main()
{
    your_asm_fn();
}
```

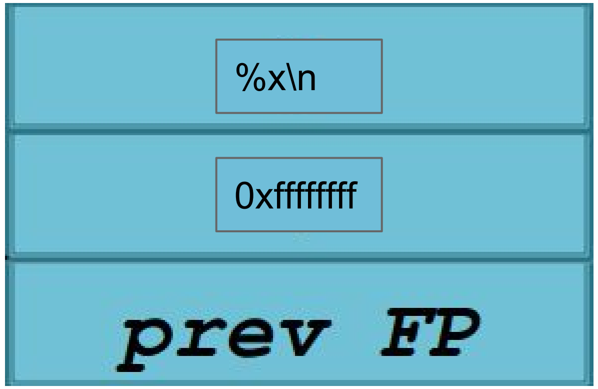
```
.global your_asm_fn
.section .text

your_asm_fn:

push    %ebp
mov     %esp,%ebp

push    $0xffffffff
push    $0xa7825


call    printf

leave
ret
```

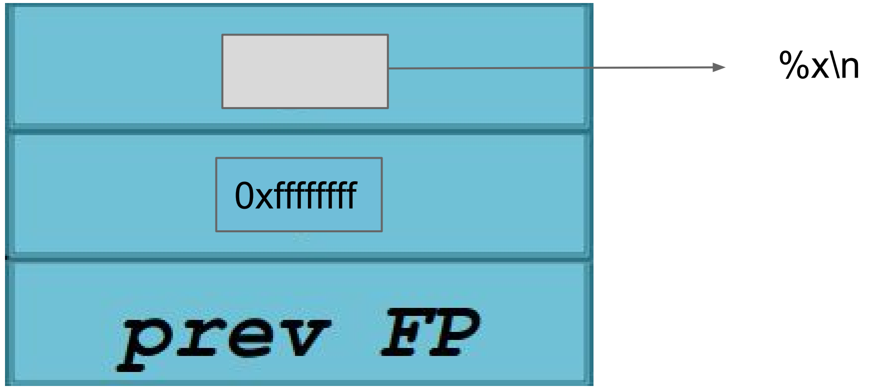
```c
#include <stdio.h>

void output(int candy)
{
    printf("%x\n",candy);
}


void main()
{
    your_asm_fn();
}
```

```asm
.global your_asm_fn
.section .text

your_asm_fn:

push    %ebp
mov     %esp,%ebp

push    $0xffffffff
push    $0xa7825


call    printf

leave
ret
```



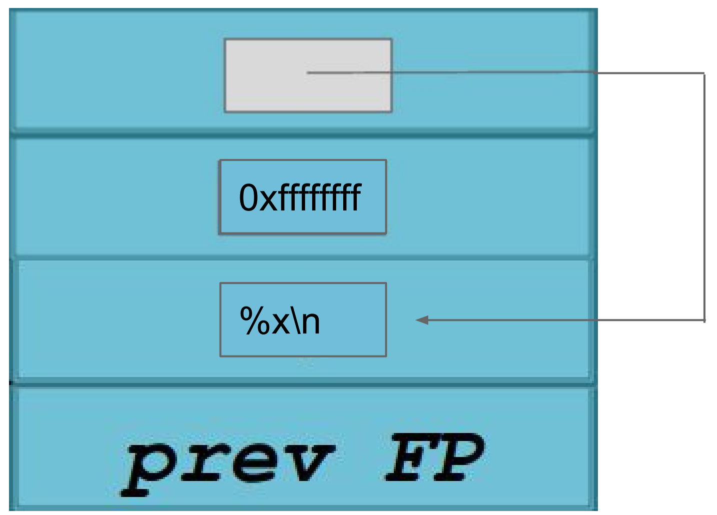
```
ubuntu@ubuntu:~/Desktop/cp1_discussion_programs$ ./demo
Segmentation fault (core dumped)
```

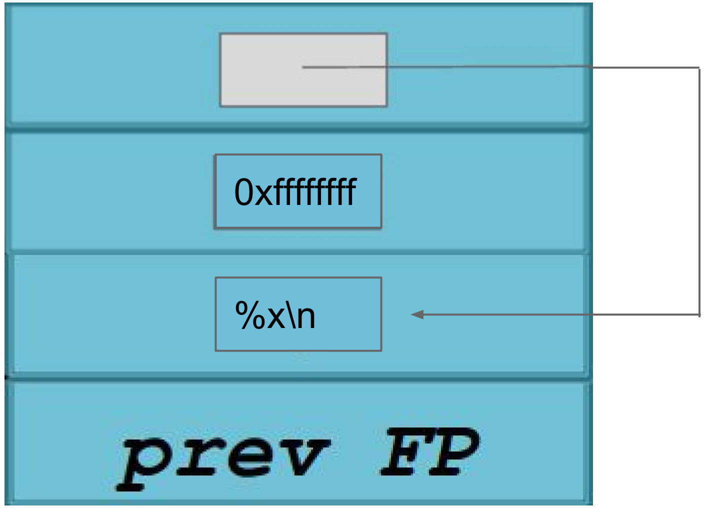

%x\n
0xffffffff
prev FP

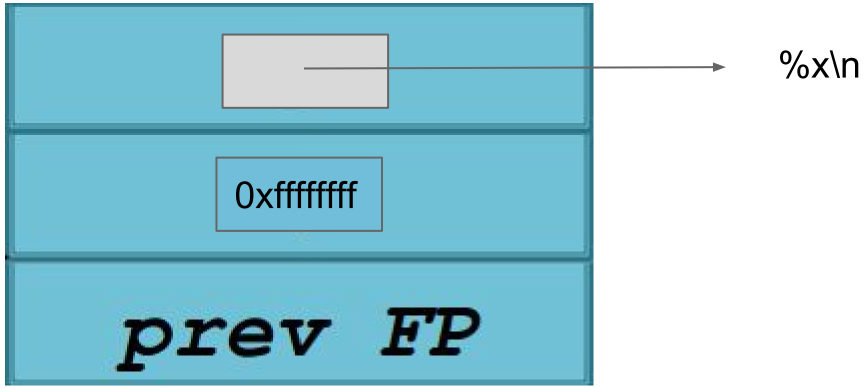

%x\n
0xffffffff
prev FP

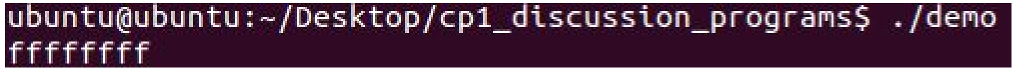
```c
#include <stdio.h>

void output(int candy)
{
    printf("%x\n",candy);
}


void main()
{
    your_asm_fn();
}
```

```asm
.global your_asm_fn
.section .text

your_asm_fn:

push    %ebp
mov     %esp,%ebp

push    $0xa7825
mov     %esp,%eax

push    $0xffffffff
push    %eax


call    printf

leave
ret
```


0xffffffff
%x\n
prev FP







```
ubuntu@ubuntu:~/Desktop/cp1_discussion_programs$ ./demo
ffffffff
```

Wait...

0xa7825 == %x\n ?????

0x0a == \n

0x78 == x

0x25 == %

Endianness

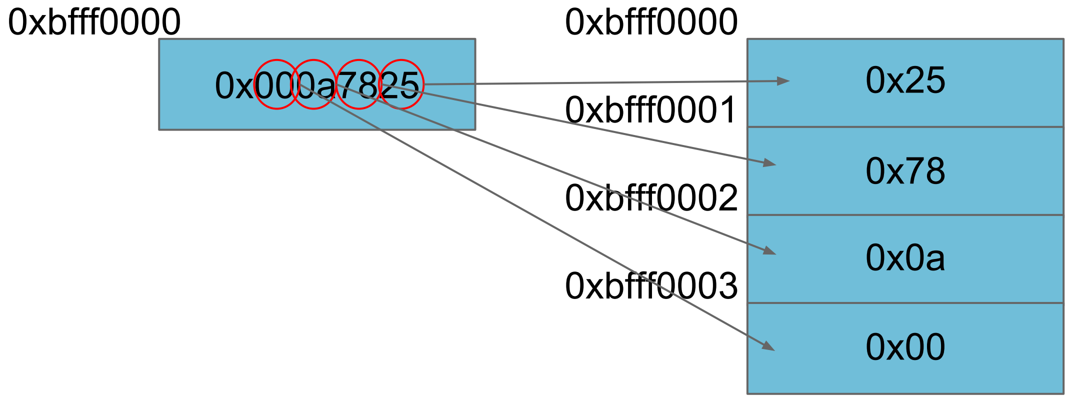
Byte order for x86 is little endian

Read from top of stack to bottom

(low memory to high memory)

Whatever gets read first is little -> small -> least significant byte

push $0xa7825 = push $0x000a7825



0x25 == %

0x78 == x

0x0a == \n

0xa7825 == %x\n

1.1.5 Introduction to Linux function calls (4 points)

Your goal for this practice is to invoke a system call through `int 0x80` to open up a shell.

Tips:

1. Use the system call `sys_execve` with the correct arguments.
2. The funtion signature of `sys_execve` in C:
   `int execve(const char *filename, char *const argv[],char *const envp[]);`
3. Instead of passing the arguments through the stack, arguments should be put into registers for system calls.
4. The system call number should be placed in register `eax`.
5. The arguments for system calls should be placed in `ebx`, `ecx`, `edx`, `esi`, `edi`, and `ebp` in order
6. To start a shell, the first argument (filename) should be a string that contains something like /bin/sh.
7. Reading Linux man pages may help.
8. Some arguments may need to be terminated with a null character/pointer.

What to submit Submit your x86 assembly code in `1.1.5.S`.

Shellcode TODO list

```
0xbffffda0: "/bin/sh\x00"
0xbffffda8: "\xa0\xfd\xff\xbf\x00\x00\x00\x00"

            11(0xb)
%eax = 1~~3~~  (sys_execve)
%ebx = 0xbffffda0          # "/bin/sh"
%ecx = 0xbffffda8          # argv
%edx = 0x00                # NULL
int 0x80
```

Prototype shellcode

```
mov     $0xb,%eax              #sys_execve
mov     $0xbffffba0,%ebx       #addr of some mem
lea     8(%ebx),%ecx           #ecx=ebx+~~12~~(argv)
xorl    %edx,%edx              #edx=NULL +8
movl    $0x6e69622f,(%ebx)     #"/bin"
movl    $0x68732f,4(%ebx)      #"/sh\x00"
mov     %ebx,(%ecx)            #argv[0]="/bin/sh"
mov     %edx,4(%ecx)           #argv[1]=NULL
int     $0x80                  #sys_execve()
```

(assume 0xbffffba0 is on the stack for now and is readable/writeable)

Reading Materials:

GCC Assembly

http://www.ibiblio.org/gferg/ldp/GCC-Inline-Assembly-HOWTO.html#s3

https://courses.engr.illinois.edu/ece391/references/doc-x86-asm.pdf