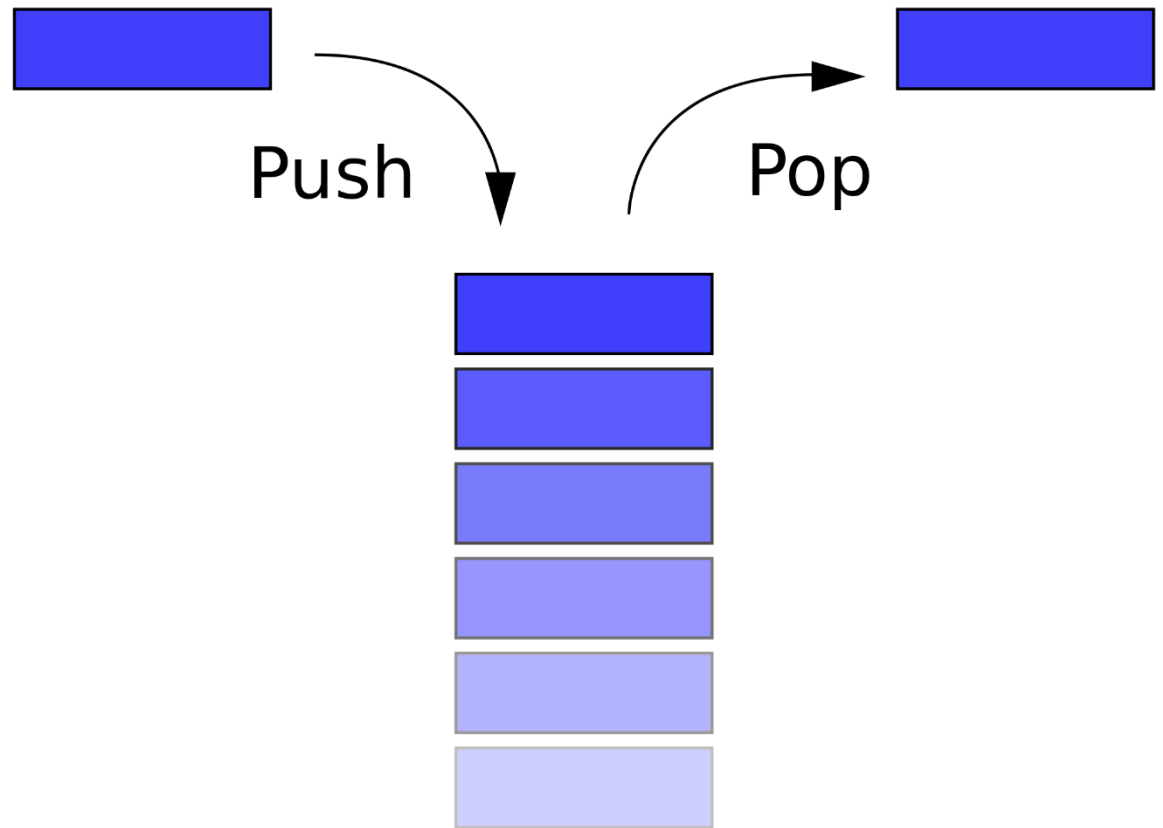
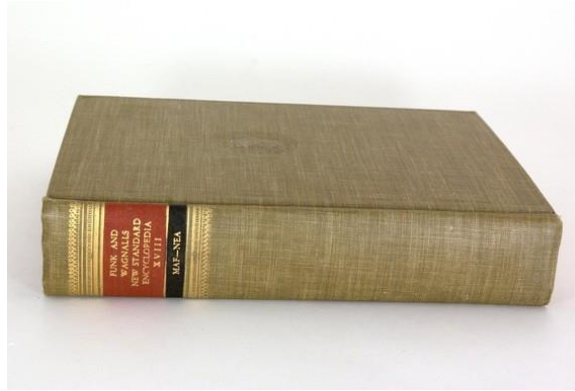


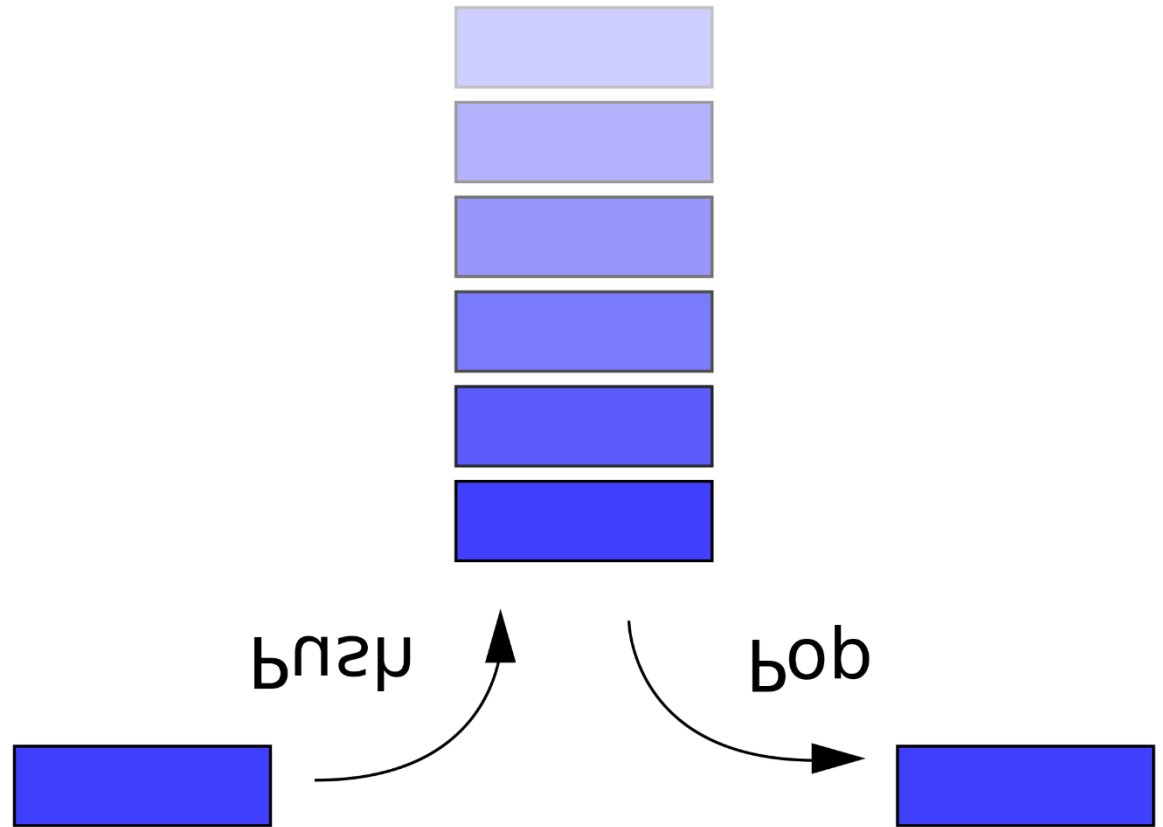
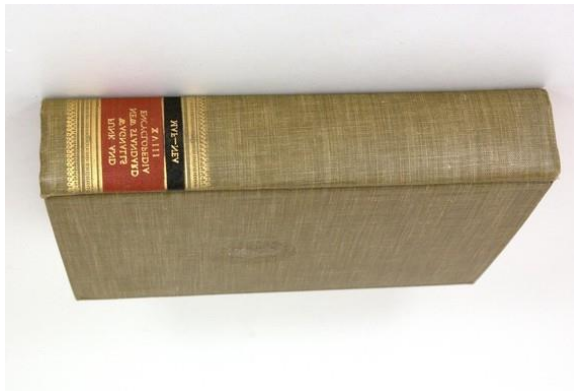
X86 Stack Called Function POV

Computer Systems Section 3.7

The Stack (as we learned in CS-120)



The x86 stack



x86 Stack

Reg	Value
esp	xFFFF FFF4
eax	x0000 000E

%esp points at
bottom of stack

- Memory above %esp is in use
- Memory below %esp is available

Memory	
Address	Value
xFFFF FFFC	
xFFFF FFF8	x0000 0004
xFFFF FFF4	x0000 0003
xFFFF FFF0	
xFFFF FFEC	
x0000 0004	
x0000 0000	

Top of stack at
high memory

x86 Stack

Reg	Value
esp	xFFFF FFF4
eax	x0000 000E



Memory	
Address	Value
xFFFF FFFC	
xFFFF FFF8	x0000 0004
xFFFF FFF4	x0000 0003
xFFFF FFF0	
xFFFF FFEC	
x0000 0004	
x0000 0000	

push %eax

x86 Stack

Reg	Value
esp	xFFFF FFF0
eax	x0000 000E

Memory	
Address	Value
xFFFF FFFC	
xFFFF FFF8	x0000 0004
xFFFF FFF4	x0000 0003
xFFFF FFF0	x0000 000E
xFFFF FFEC	
x0000 0004	
x0000 0000	

push %eax

subl \$4,%esp
movl %eax,(%esp)

x86 Stack

Reg	Value
esp	xFFFF FFF4
eax	x0000 000E
ebx	x0000 000E

Memory	
Address	Value
xFFFF FFFC	
xFFFF FFF8	x0000 0004
xFFFF FFF4	x0000 0003
xFFFF FFF0	x0000 000E
xFFFF FFEC	
x0000 0004	
x0000 0000	

pop %ebx

movl (%esp),%ebx
addl \$4,%esp

Stack Frame

Reg	Value
ebp	xFFFF FFF0
esp	xFFFF FFE0
eax	x0000 000E



Previous Frame
directly above
current frame

Current Frame
between word at
%ebp and %esp

What's In a Stack Frame?

- Caller's frame info (pointer to top of caller's frame)
- Space for Local Variable Values
- Space for saved state
- Space for arguments to lower level functions
- Return address (when calling functions)

How Big is a Stack Frame?

Info	Size
Caller's frame info	4 bytes
Local Variables	? (different for each function)
Saved State	? (different for each function)
Arguments	? (depends on who is called)
Return Address	4 bytes (if needed)
Total	4+???

Frame Contents

Reg	Value
ebp	xFFFF FFF0
esp	xFFFF FFD0
eax	x0000 000E



Caller's
base pointer

Local Variables

Saved Regs

Arguments

Return Address

Frame Offsets

Reg	Value
ebp	xFFFF FFF0
esp	xFFFF FFD0
eax	x0000 000E

Memory		
Labels	Address	Value
8(%ebp)	xFFFF FFF8	x0000 0005
4(%ebp)	xFFFF FFF4	x0000 0004
(%ebp)	xFFFF FFF0	xFFFF FFF8
-4(%ebp)	xFFFF FFEC	x0000 0003
-8(%ebp)	xFFFF FFE8	x0000 0003
	xFFFF FFE4	x0000 00C4
	xFFFF FFE0	x0000 0003
8(%esp)	xFFFF FFDC	x0000 0004
4(%esp)	xFFFF FFD8	x0000 0003
(%esp)	xFFFF FFD0	x080483c8
	x0000 0000	

Parameters

Caller's
base pointer

Local Variables

Saved Regs

Arguments

Return Address

When I am called...

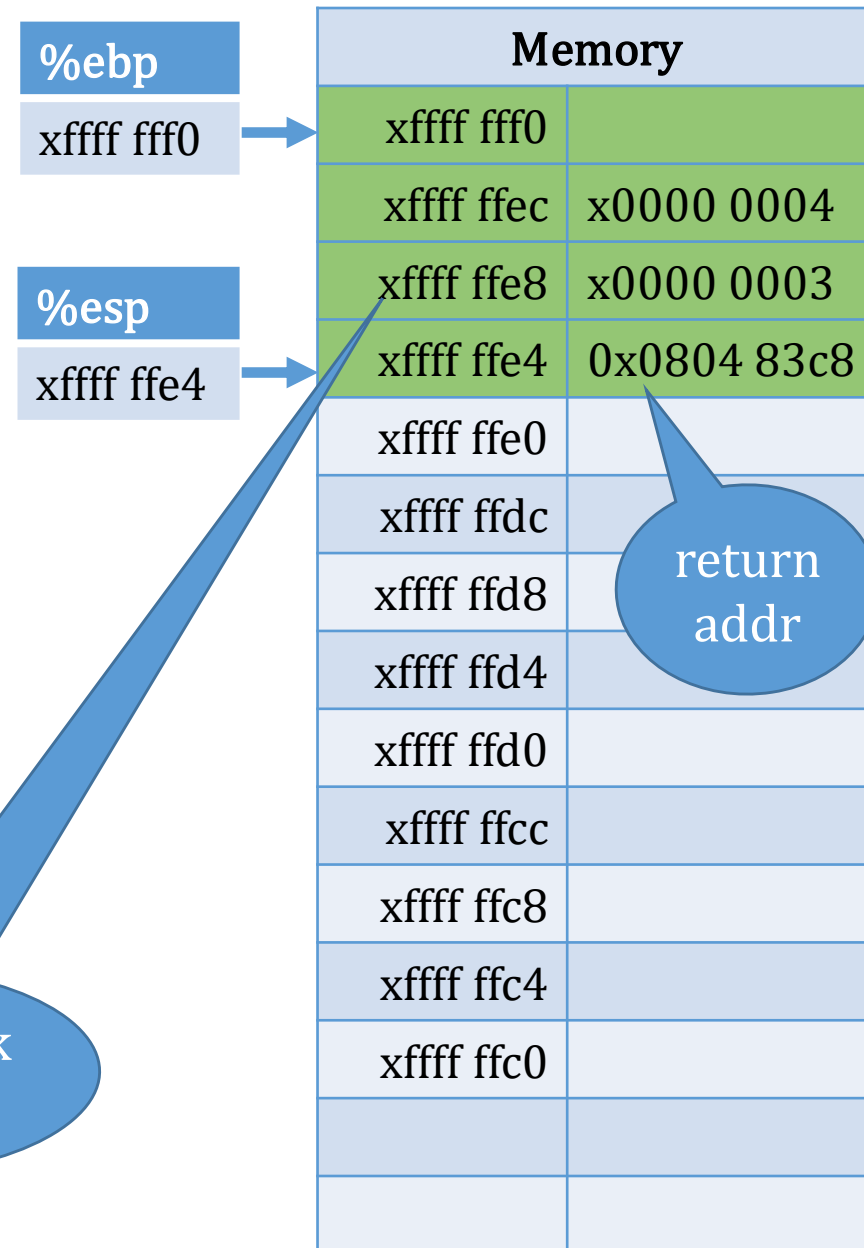
- My caller's stack frame is still active
- I need to save information about my callers frame
- I need to create my own stack frame

Stack when myfunc called

myfunc:

```
int myfunc(int a,int b) {
    int c;
    c = a + 3;
    c = c + b;
    return c;
}
```

```
pushl    %ebp
movl     %esp, %ebp
subl     $16, %esp
movl     8(%ebp), %eax
addl     $3, %eax
movl     %eax, -4(%ebp)
movl     12(%ebp), %eax
addl     %eax, -4(%ebp)
movl     -4(%ebp), %eax
leave
ret
```

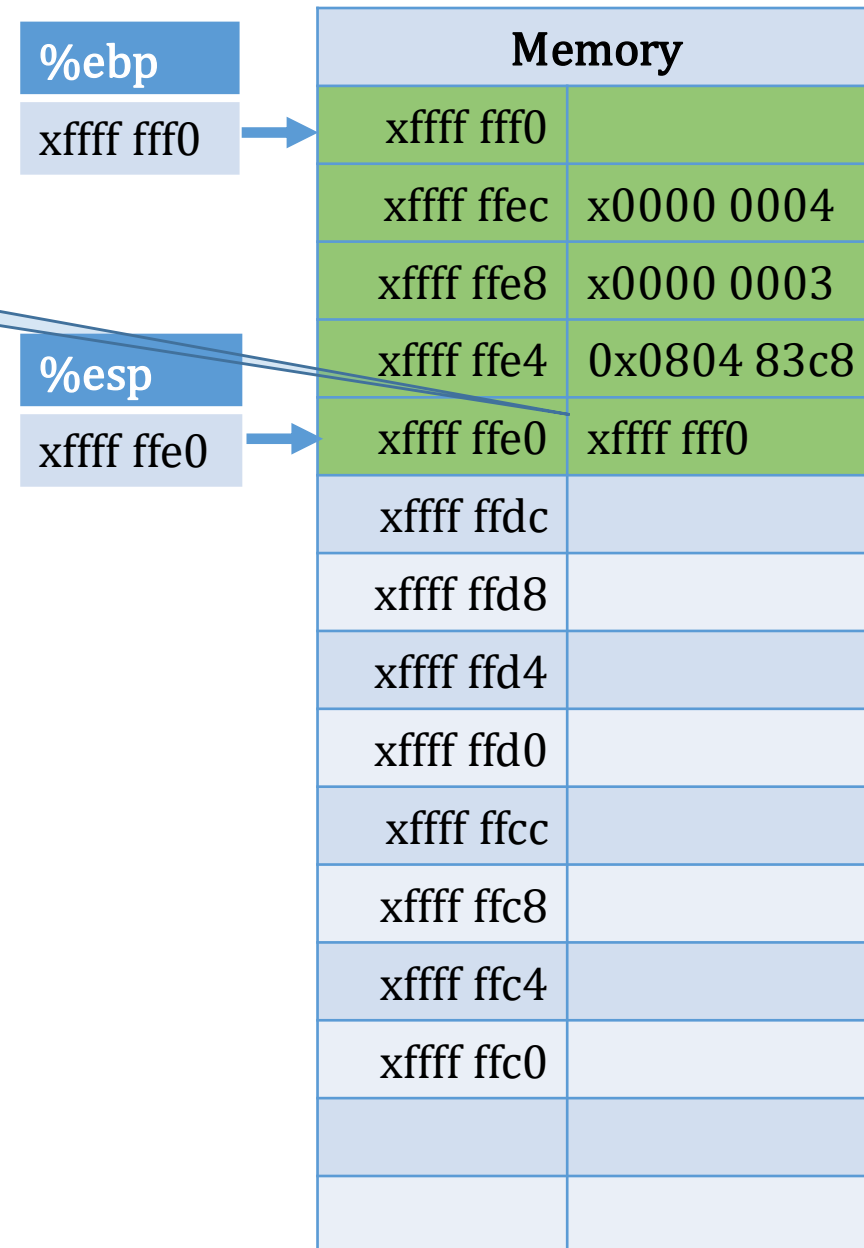


Save caller's base pointer

myfunc:

```
int myfunc(int a,int b) {
    int c;
    c = a + 3;
    c = c + b;
    return c;
}
```

```
    pushl %ebp
    movl  %esp, %ebp
    subl  $16, %esp
    movl  8(%ebp), %eax
    addl  $3, %eax
    movl  %eax, -4(%ebp)
    movl  12(%ebp), %eax
    addl  %eax, -4(%ebp)
    movl  -4(%ebp), %eax
    leave
    ret
```

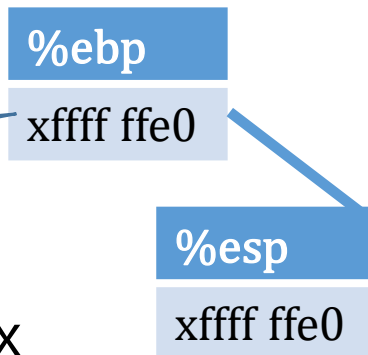


Create new frame

myfunc:

```
int myfunc(int a,int b) {
    int c;
    c = a + 3;
    c = c + b;
    return c;
}
```

```
pushl %ebp
movl %esp, %ebp
subl $16, %esp
movl 8(%ebp), %eax
addl $3, %eax
movl %eax, -4(%ebp)
movl 12(%ebp), %eax
addl %eax, -4(%ebp)
movl -4(%ebp), %eax
leave
ret
```



Memory	
ffffffff0	
fffffffec	x0000 0004
fffffffe8	x0000 0003
fffffffe4	0x0804 83c8
fffffffe0	ffffffff0
fffffffdc	
fffffffd8	
fffffffd4	
fffffffd0	
fffffffcc	
fffffffc8	
fffffffc4	
fffffffc0	

Create new frame (part 2)

myfunc:

```
int myfunc(int a,int b) {
    int c;
    c = a + 3;
    c = c + b;
    return c;
}
```

```
pushl %ebp
movl %esp, %ebp
subl $16, %esp
movl 8(%ebp), %eax
addl $3, %eax
movl %eax, -4(%ebp)
movl 12(%ebp), %eax
addl %eax, -4(%ebp)
movl -4(%ebp), %eax
leave
ret
```

%ebp

xxxxffe0

%esp

xxxxffd0

Memory

xxxxfff0	
xxxxffec	x0000 0004
xxxxffe8	x0000 0003
xxxxffe4	0x0804 83c8
xxxxffe0	xxxxfff0
xxxxffdc	
xxxxffd8	
xxxxffd4	
xxxxffd0	
xxxxffcc	
xxxxffc8	
xxxxffc4	
xxxxffc0	

Myfunc Execution
here.

Save return value

```
int myfunc(int a,int b) {
    int c;
    c = a + 3;
    c = c + b;
    return c;
}
```

myfunc:

```
pushl %ebp
movl %esp, %ebp
subl $16, %esp
movl 8(%ebp), %eax
addl $3, %eax
movl %eax, -4(%ebp)
movl 12(%ebp), %eax
addl %eax, -4(%ebp)
movl -4(%ebp), %eax
leave
ret
```

%eax holds
return
value

%ebp

xxxxx ffe0

%esp

xxxxx ffd0

%eax

x0000 000a

Memory

xxxxx fff0	
xxxxx ffec	x0000 0004
xxxxx ffe8	x0000 0003
xxxxx ffe4	0x0804 83c8
xxxxx ffe0	xxxxx fff0
xxxxx ffdc	x0000 000a
xxxxx ffd8	
xxxxx ffd4	
xxxxx ffd0	
xxxxx ffcc	
xxxxx ffc8	
xxxxx ffc4	
xxxxx ffc0	

Give up my frame

myfunc:

```
int myfunc(int a,int b) {
    int c;
    c = a + 3;
    c = c + b;
    return c;
}
```

```
    pushl    %ebp
    movl     %esp, %ebp
    subl     $16, %esp
    movl     8(%ebp), %eax
    addl     $3, %eax
    movl     %eax, -4(%ebp)
    movl     12(%ebp), %eax
    addl     %eax, -4(%ebp)
    movl     -4(%ebp), %eax
    leave
    ret
```

leave:
mov %ebp,%esp
pop %ebp

%ebp
xffffffe0

%esp
xffffffe0

%eax
x0000 000a

Memory	
xfffffff0	
ffffffec	x0000 0004
ffffffe8	x0000 0003
ffffffe4	0x0804 83c8
ffffffe0	xfffffff0
ffffffd8	x0000 000a
ffffffd4	
ffffffd0	
ffffffcc	
ffffffc8	
ffffffc4	
ffffffc0	

Restore Caller's Frame

myfunc:

```
int myfunc(int a,int b) {
    int c;
    c = a + 3;
    c = c + b;
    return c;
}
```

```
pushl    %ebp
movl     %esp, %ebp
subl     $16, %esp
movl     8(%ebp), %eax
addl     $3, %eax
movl     %eax, -4(%ebp)
movl     12(%ebp), %eax
addl     %eax, -4(%ebp)
movl     -4(%ebp), %eax
leave
ret
```

leave:
mov %ebp,%esp
pop %ebp

%ebp

xffff fff0

%esp

ffff ffe4

%eax

x0000 000a

Memory

xffff fff0	
xffff ffec	x0000 0004
xffff ffe8	x0000 0003
ffff ffe4	0x0804 83c8
ffff ffe0	ffff fff0
ffff ffdc	x0000 000a
ffff ffd8	
ffff ffd4	
ffff ffd0	
ffff ffcc	
ffff ff8	
ffff ffc4	
ffff ffc0	

X86 Calling Conventions (Simple)

On Entry

- Save caller's base pointer
- Create my stack frame
- Initialize local variables

On Exit

- Save return value in %eax
- Return my stack frame
- Restore caller's stack frame

Return to caller

myfunc:

```
int myfunc(int a,int b) {
    int c;
    c = a + 3;
    c = c + b;
    return c;
}
```

```
pushl    %ebp
movl     %esp, %ebp
subl     $16, %esp
movl     8(%ebp), %eax
addl     $3, %eax
movl     %eax, -4(%ebp)
movl     12(%ebp), %eax
addl     %eax, -4(%ebp)
movl     -4(%ebp), %eax
leave
```

ret

ret:
pop %eip

%ebp
xffff fff0

%esp
xffff ffe8

%eax
x0000 000a

%eip
0x0804 83c8

return
addr

Memory	
xffff fff0	
xffff ffec	x0000 0004
xffff ffe8	x0000 0003
xffff ffe4	0x0804 83c8
xffff ffe0	xffff fff0
xffff ffdc	x0000 000a
xffff ffd8	
xffff ffd4	
xffff ffd0	
xffff ffcc	
xffff ff88	
xffff ff44	
xffff ff00	