ECE 391 MT 1 Review

Patrick Marschoun, Noelle Crawford

Reminders

- Exam is **Tuesday, September 26th (7:00-9:00 pm)**
  - 1 page of notes allowed
  - Notes sheet will be collected (Make a copy!)
  - Please put your name on notes
- Keep the lab clean

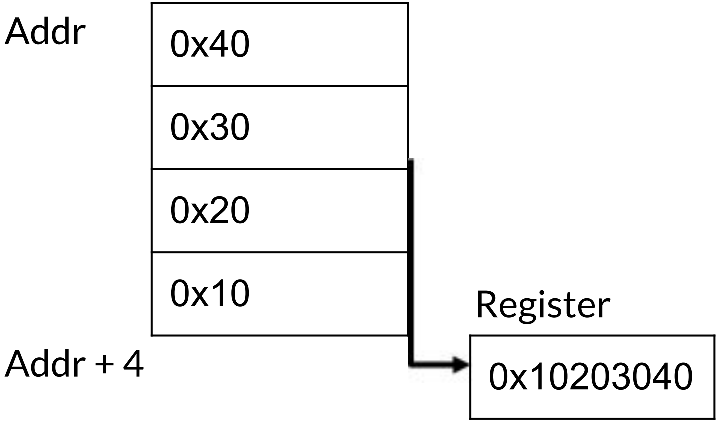
Exam Content

- Assembly
- C Calling Convention
- Synchronization
- PIC
- MP1

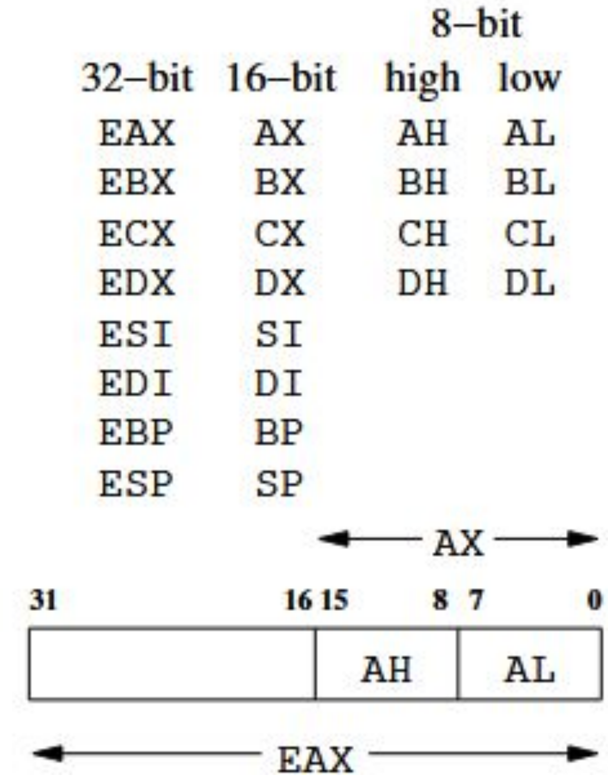
x86 Assembly

x86

- Little Endian
- Byte Addressable
- Important Registers: ESP, EBP, EAX (Kinda)
- General Purpose Registers: EBX, ECX, EDX, ESI, EDI



Registers



		8–bit	
32–bit	16–bit	high	low
EAX	AX	AH	AL
EBX	BX	BH	BL
ECX	CX	CH	CL
EDX	DX	DH	DL
ESI	SI		
EDI	DI		
EBP	BP		
ESP	SP		

x86 Operate Instructions

- ADD, SUB, NEG
- INC, DEC
- AND, OR, XOR, NOT
- SHL, SAR, SHR
- ROL, ROR

x86 Data Movement Instructions

- MOV, LEA
- Memory Addressing
    - *displacement (SR1, SR2, scale)*
      is equivalent to
      SR1 + (SR2 ✖ scale) + displacement

Question?

What’s the difference between

leal (%eax, %ebx), %ecx

and

movl (%eax, %ebx), %ecx

Answer!

LEA:

# ECX ⟵ EAX + EBX

MOV:

# ECX ⟵ M[EAX + EBX]

Question?

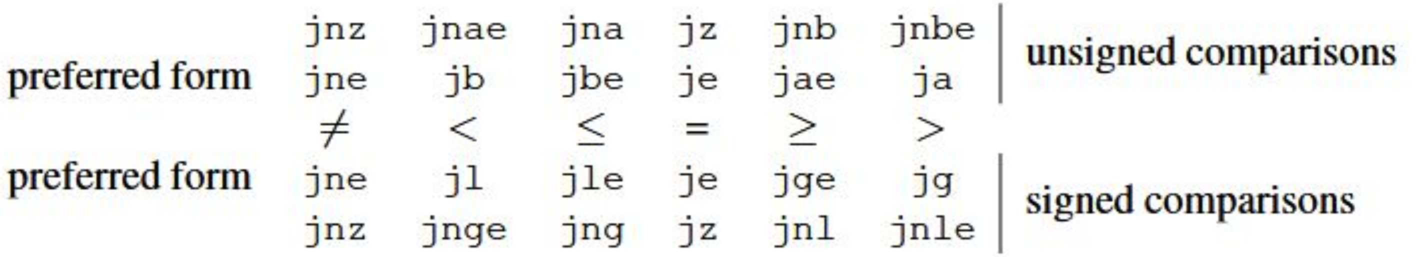
Let’s say you have an array with 8 byte elements (struct with 2 integers). EBX contains the pointer to the start of the array. You want to access the 16th element of the array, so ECX contains 15. What instruction would place the second integer of that element into EDX?

Answer!

```
movl 4(%ebx, %ecx, 8), %edx
```

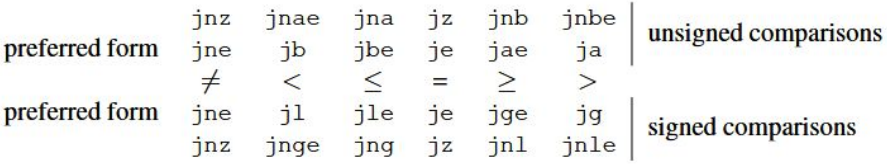
x86 Conditional Instructions

- CMP, TEST



	jnz	jnae	jna	jz	jnb	jnbe	unsigned comparisons
preferred form	jne	jb	jbe	je	jae	ja	
	≠	<	≤	=	≥	>	
preferred form	jne	jl	jle	je	jge	jg	signed comparisons
	jnz	jnge	jng	jz	jnl	jnle	

Question?



	jnz	jnae	jna	jz	jnb	jnbe	unsigned comparisons
preferred form	jne	jb	jbe	je	jae	ja	
	≠	<	≤	=	≥	>	
preferred form	jne	jl	jle	je	jge	jg	signed comparisons
	jnz	jnge	jng	jz	jnl	jnle	

You have the following line

```
cmp %ebx, %esi
```

where EBX and ESI contain signed integer values. You want to branch if EBX < ESI, what conditional jump should you use?

Answer!

JG or JNLE

cmp %ebx, %esi

Flags set based on ESI - EBX

ESI > EBX

So greater than, JG

x86 Instructions Cont

- JMP, CALL
- PUSH, POP

Question?

What’s the difference between JMP and CALL?

Answer!

JMP doesn’t change the stack

CALL pushes the EIP onto the stack

Questions about x86?

Calling Convention

C Calling Convention

- Caller vs Callee
- Caller Saved Registers
    - EAX, ECX, EDX, EFLAGS
- Callee Saved Registers
    - EBX, ESI, EDI

Example

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

int main(){
	return my_function(3, 5);
}
```

Example

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

int main(){
    return my_function(3, 5);
}
```

```
my_function:
        pushl %ebp
        movl %esp, %ebp
        movl 8(%ebp), %eax
        movl 12(%ebp), %ebx
        addl %ebx, %eax
        leave
        ret

main:
        pushl %ebp
        movl %esp, %ebp
        push $5
        push $3
        call my_function
        addl $8, %esp
        leave
        ret
```

Example

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

int main(){
        return my_function(3, 5);
}
```

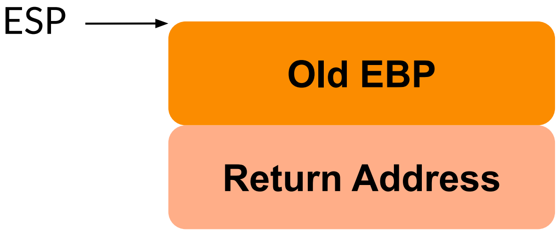
```
my_function:
        pushl %ebp
        movl %esp, %ebp
        movl 8(%ebp), %eax
        movl 12(%ebp), %ebx
        addl %ebx, %eax
        leave
        ret

main:
        pushl %ebp
        movl %esp, %ebp
        push $5
        push $3
        call my_function
        addl $8, %esp
        leave
        ret
```

Return Address

Example

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

int main(){
        return my_function(3, 5);
}
```

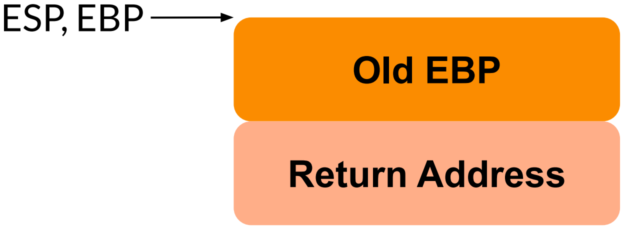
```
my_function:
        pushl %ebp
        movl %esp, %ebp
        movl 8(%ebp), %eax
        movl 12(%ebp), %ebx
        addl %ebx, %eax
        leave
        ret

main:
        pushl %ebp
        movl %esp, %ebp
        push $5
        push $3
        call my_function
        addl $8, %esp
        leave
        ret
```



Example

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

int main(){
	return my_function(3, 5);
}
```

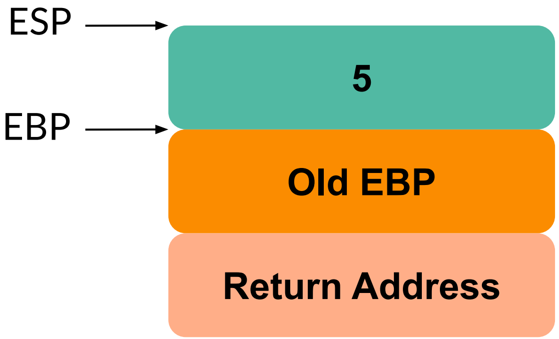
```
my_function:
	pushl %ebp
	movl %esp, %ebp
	movl 8(%ebp), %eax
	movl 12(%ebp), %ebx
	addl %ebx, %eax
	leave
	ret

main:
	pushl %ebp
	movl %esp, %ebp
	push $5
	push $3
	call my_function
	addl $8, %esp
	leave
	ret
```



Example

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

int main(){
        return my_function(3, 5);
}
```

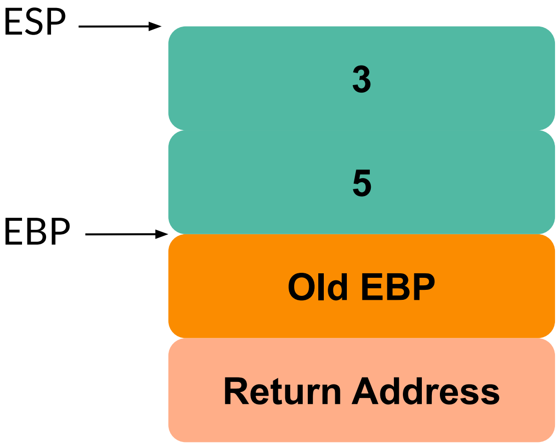
```
my_function:
        pushl %ebp
        movl %esp, %ebp
        movl 8(%ebp), %eax
        movl 12(%ebp), %ebx
        addl %ebx, %eax
        leave
        ret

main:
        pushl %ebp
        movl %esp, %ebp
        push $5
        push $3
        call my_function
        addl $8, %esp
        leave
        ret
```



Example

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

int main(){
        return my_function(3, 5);
}
```

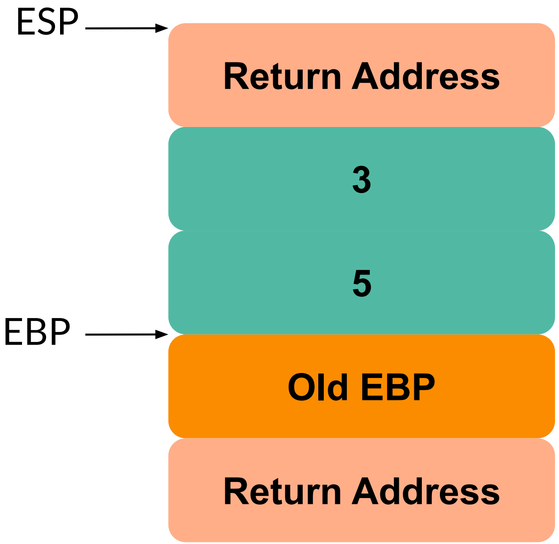
```
my_function:
        pushl %ebp
        movl %esp, %ebp
        movl 8(%ebp), %eax
        movl 12(%ebp), %ebx
        addl %ebx, %eax
        leave
        ret

main:
        pushl %ebp
        movl %esp, %ebp
        push $5
        push $3
        call my_function
        addl $8, %esp
        leave
        ret
```



Example

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

int main(){
    return my_function(3, 5);
}
```

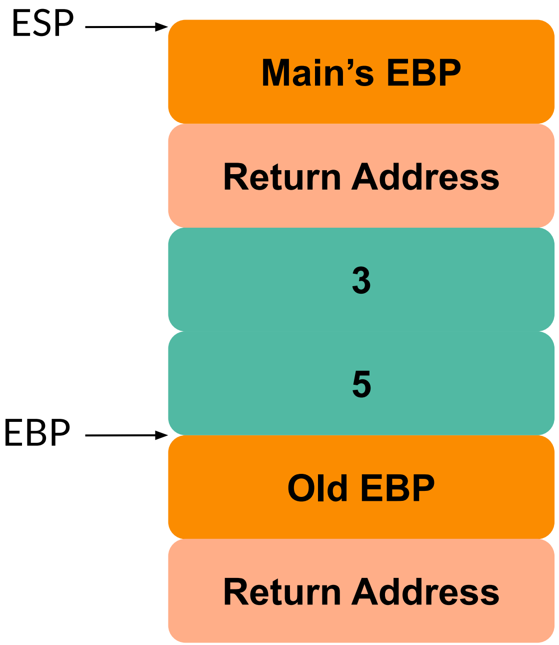
```
my_function:
        pushl %ebp
        movl %esp, %ebp
        movl 8(%ebp), %eax
        movl 12(%ebp), %ebx
        addl %ebx, %eax
        leave
        ret

main:
        pushl %ebp
        movl %esp, %ebp
        push $5
        push $3
        call my_function
        addl $8, %esp
        leave
        ret
```



Example

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

int main(){
      return my_function(3, 5);
}
```

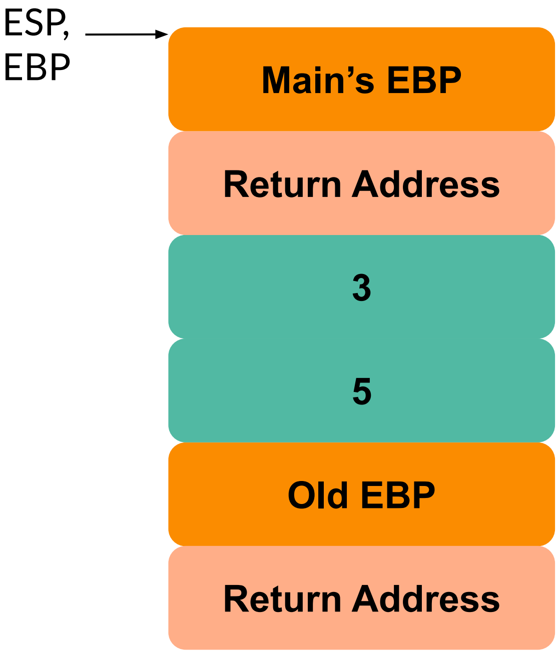
```
my_function:
        pushl %ebp
        movl %esp, %ebp
        movl 8(%ebp), %eax
        movl 12(%ebp), %ebx
        addl %ebx, %eax
        leave
        ret

main:
        pushl %ebp
        movl %esp, %ebp
        push $5
        push $3
        call my_function
        addl $8, %esp
        leave
        ret
```



Example

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

int main(){
	return my_function(3, 5);
}
```

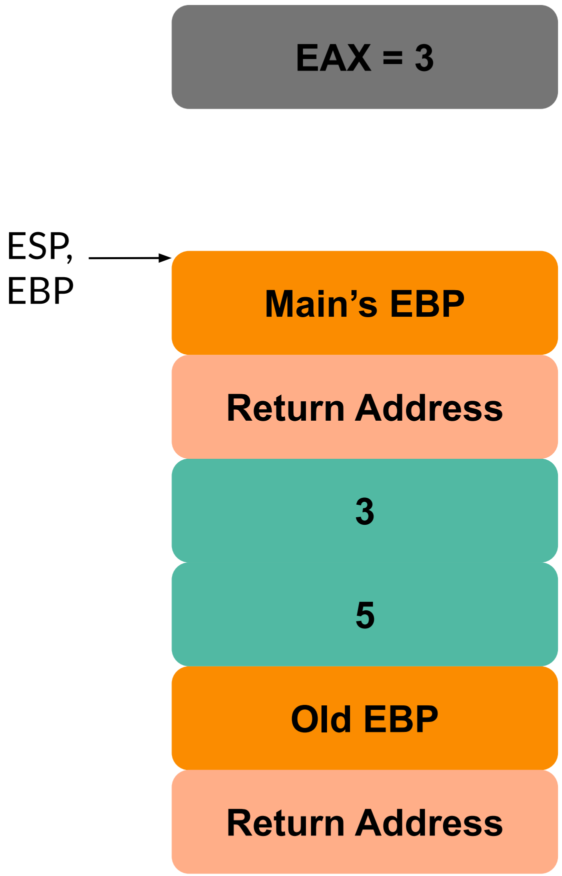
```
my_function:
	pushl %ebp
	movl %esp, %ebp
	movl 8(%ebp), %eax
	movl 12(%ebp), %ebx
	addl %ebx, %eax
	leave
	ret

main:
	pushl %ebp
	movl %esp, %ebp
	push $5
	push $3
	call my_function
	addl $8, %esp
	leave
	ret
```



Example

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

int main(){
      return my_function(3, 5);
}
```

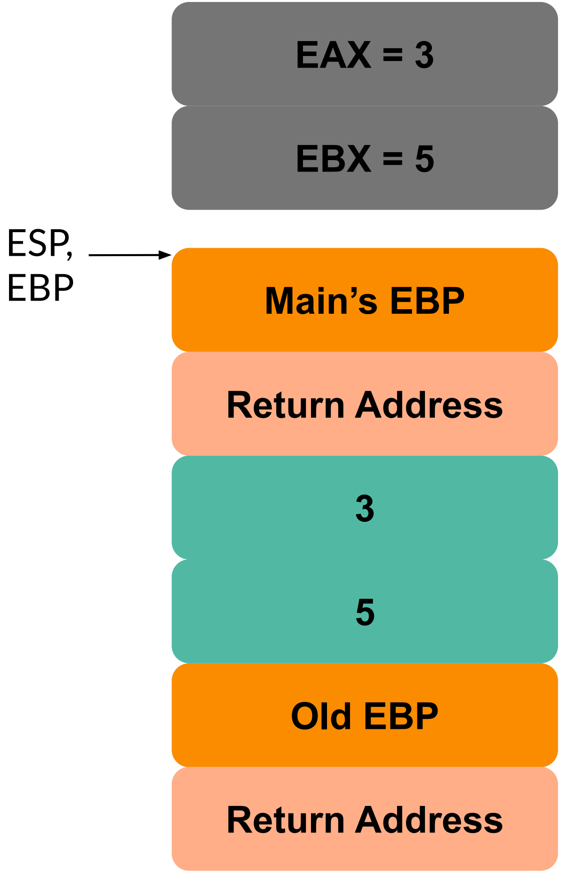
```
my_function:
        pushl %ebp
        movl %esp, %ebp
        movl 8(%ebp), %eax
        movl 12(%ebp), %ebx
        addl %ebx, %eax
        leave
        ret

main:
        pushl %ebp
        movl %esp, %ebp
        push $5
        push $3
        call my_function
        addl $8, %esp
        leave
        ret
```



Example

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

int main(){
        return my_function(3, 5);
}
```

```
my_function:
        pushl %ebp
        movl %esp, %ebp
        movl 8(%ebp), %eax
        movl 12(%ebp), %ebx
        addl %ebx, %eax
        leave
        ret

main:
        pushl %ebp
        movl %esp, %ebp
        push $5
        push $3
        call my_function
        addl $8, %esp
        leave
        ret
```



EAX = 3

EBX = 5

ESP, EBP

Main's EBP

Return Address

3

5

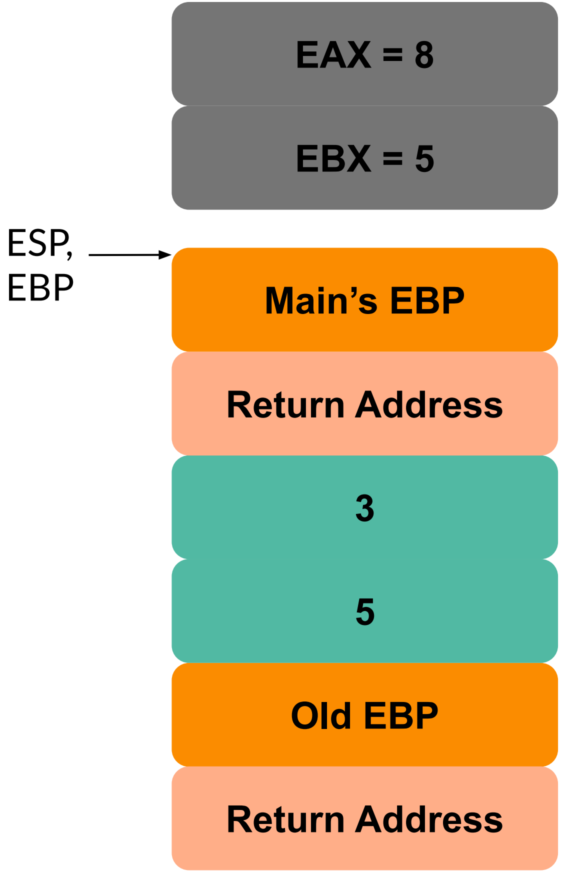
Old EBP

Return Address

Example

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

int main(){
        return my_function(3, 5);
}
```

```
my_function:
        pushl %ebp
        movl %esp, %ebp
        movl 8(%ebp), %eax
        movl 12(%ebp), %ebx
        addl %ebx, %eax
        leave
        ret

main:
        pushl %ebp
        movl %esp, %ebp
        push $5
        push $3
        call my_function
        addl $8, %esp
        leave
        ret
```



EAX = 8

EBX = 5

ESP, EBP

Main’s EBP

Return Address

3

5

Old EBP

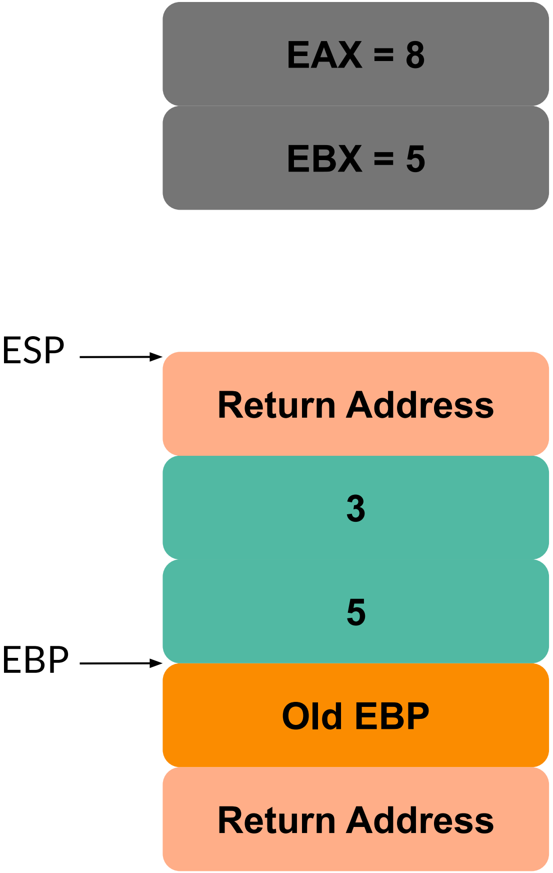
Return Address

Example

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

int main(){
        return my_function(3, 5);
}
```

```
my_function:
        pushl %ebp
        movl %esp, %ebp
        movl 8(%ebp), %eax
        movl 12(%ebp), %ebx
        addl %ebx, %eax
        leave
        ret

main:
        pushl %ebp
        movl %esp, %ebp
        push $5
        push $3
        call my_function
        addl $8, %esp
        leave
        ret
```

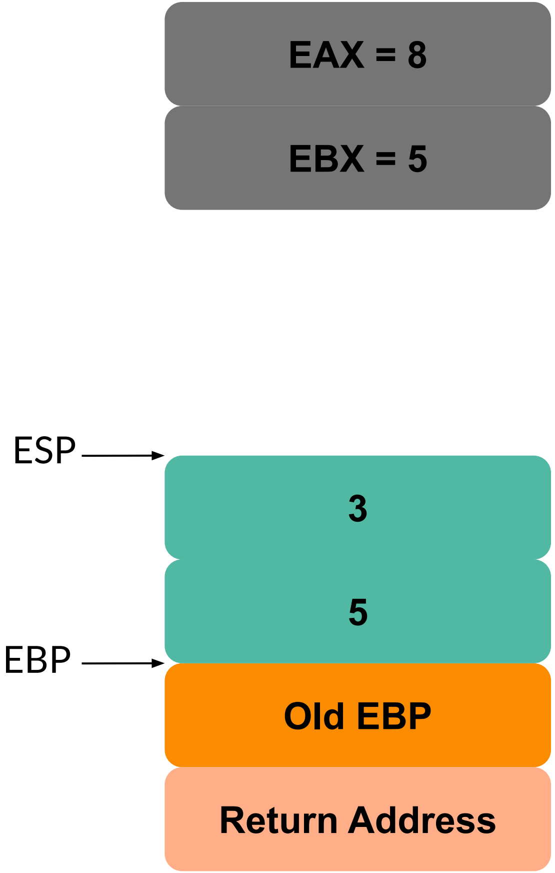
leave



Example

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

int main(){
    return my_function(3, 5);
}
```

```
my_function:
        pushl %ebp
        movl %esp, %ebp
        movl 8(%ebp), %eax
        movl 12(%ebp), %ebx
        addl %ebx, %eax
        leave
        ret

main:
        pushl %ebp
        movl %esp, %ebp
        push $5
        push $3
        call my_function
        addl $8, %esp
        leave
        ret
```



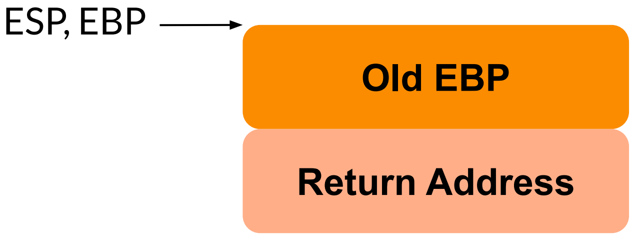
Example

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

int main(){
	return my_function(3, 5);
}
```

```
my_function:
	pushl %ebp
	movl %esp, %ebp
	movl 8(%ebp), %eax
	movl 12(%ebp), %ebx
	addl %ebx, %eax
	leave
	ret

main:
	pushl %ebp
	movl %esp, %ebp
	push $5
	push $3
	call my_function
	addl $8, %esp
	leave
	ret
```

EAX = 8

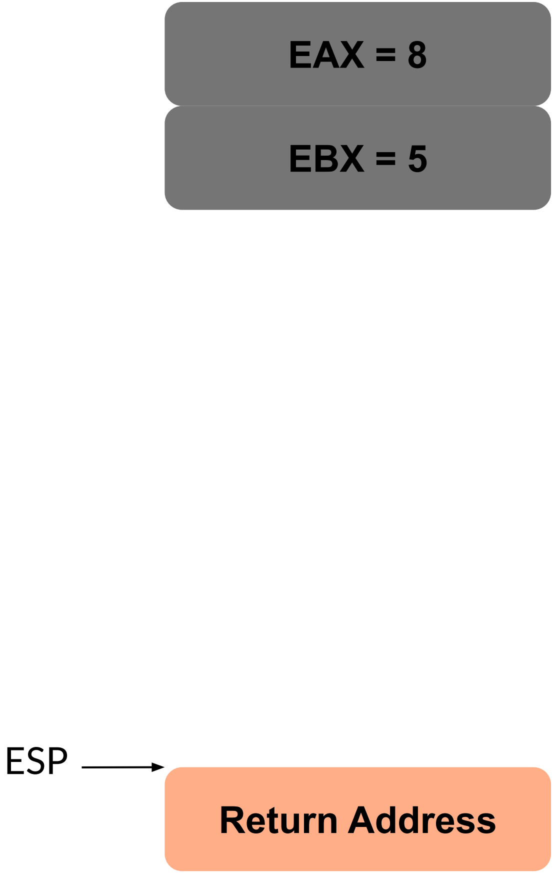
EBX = 5



Example

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

int main(){
        return my_function(3, 5);
}
```

```
my_function:
        pushl %ebp
        movl %esp, %ebp
        movl 8(%ebp), %eax
        movl 12(%ebp), %ebx
        addl %ebx, %eax
        leave
        ret

main:
        pushl %ebp
        movl %esp, %ebp
        push $5
        push $3
        call my_function
        addl $8, %esp
        leave
        ret
```



EAX = 8

EBX = 5

ESP

Return Address

Example

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

int main(){
        return my_function(3, 5);
}
```

```
my_function:
        pushl %ebp
        movl %esp, %ebp
        movl 8(%ebp), %eax
        movl 12(%ebp), %ebx
        addl %ebx, %eax
        leave
        ret

main:
        pushl %ebp
        movl %esp, %ebp
        push $5
        push $3
        call my_function
        addl $8, %esp
        leave
        ret
```

EAX = 8

EBX = 5

ESP →

What’s wrong with this code?

Callee Save Registers

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

int main(){
	return my_function(3, 5);
}
```

```
my_function:
	pushl %ebp
	movl %esp, %ebp
	pushl %ebx
	movl 8(%ebp), %eax
	movl 12(%ebp), %ebx
	addl %ebx, %eax
	popl %ebx
	leave
	ret
```

```
main:
	pushl %ebp
	movl %esp, %ebp
	push $5
	push $3
	call my_function
	addl $8, %esp
	leave
	ret
```

Questions?

Synchronization

Why do we need synchronization

- Protect shared resources (many programs using the same data)
- Prevents interrupts from corrupting data.
- Race conditions where multiple programs are writing to same variable, causes undefined behavior.
- Critical section, must run to completion.

Solution?

- Locks!
- Spinlock - “spins” until can acquire lock, used for short critical sections
- Semaphores - thread goes to sleep until lock can be acquired
- Read/Write - prevent’s writer starvation
- Volatile Variables - Compiler will always re-load variable
    - Used in loops if variable not directly changed

Synchronization with Interrupts

- void spin_lock does not disable interrupts by itself
  - Solution: spin_lock_irqsave -> Also saves flags

- Clear interrupts before acquiring a lock
  - Why? If acquire spinlock before clearing interrupts, interrupt could occur that needs spinlock, deadlock.
  - Set interrupts after releasing lock

Example

- BAD!

  ```
  spin_lock(lock)
  CLI
  … critical section …
  STI
  spin_unlock(lock)
  ```

Example

spin_lock(lock)

CLI

… critical section …

STI

spin_unlock(lock)

Example

INT $0x80

Example

```
spin_lock(lock)
CLI
… critical section …
STI
spin_unlock(lock)
```

Interrupt Handler:

… code …
spin_lock(lock)
… code …

What will happen here?

Deadlock!

```
spin_lock(lock)
CLI
… critical section …
STI
spin_unlock(lock)
```

Interrupt Handler:

… code …
spin_lock(lock)
… code …

How to fix deadlock?

- Let's prevent deadlock by releasing locks if we aren't able to do anything with it.
- Eg: If we need two locks, release first one if we aren't able to acquire the second one as well.
- What could still happen?

Livelock!

Program_A:

```
while(true):
    spin_lock(lock_1)
    spin_trylock(lock_2)
    if not have both locks:
        spin_unlock(lock_1)
        continue
    break
```

Program_B:

```
while(true):
    spin_lock(lock_2)
    spin_trylock(lock_1)
    if not have both locks:
        spin_unlock(lock_2)
        continue
    break
```

PA	PB
L1	L2

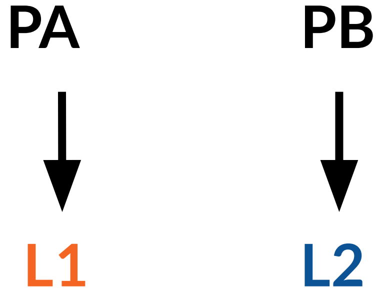
Livelock!

Program_A:

```
while(true):
    spin_lock(lock_1)
    spin_trylock(lock_2)
    if not have both locks:
        spin_unlock(lock_1)
        continue
    break
```

Program_B:

```
while(true):
    spin_lock(lock_2)
    spin_trylock(lock_1)
    if not have both locks:
        spin_unlock(lock_2)
        continue
    break
```



PA

PB

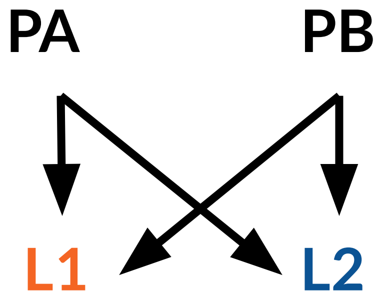
L1

L2

Livelock!

```
Program_A:

while(true):
    spin_lock(lock_1)
    spin_trylock(lock_2)
    if not have both locks:
        spin_unlock(lock_1)
        continue
    break
```

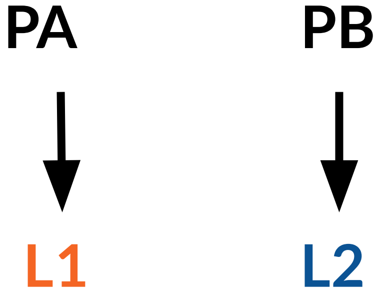
```
Program_B:

while(true):
    spin_lock(lock_2)
    spin_trylock(lock_1)
    if not have both locks:
        spin_unlock(lock_2)
        continue
    break
```



Livelock!

```
Program_A:

while(true):
    spin_lock(lock_1)
    spin_trylock(lock_2)
    if not have both locks:
        spin_unlock(lock_1)
        continue
    break
```

```
Program_B:

while(true):
    spin_lock(lock_2)
    spin_trylock(lock_1)
    if not have both locks:
        spin_unlock(lock_2)
        continue
    break
```



Livelock!

Program_A:

```
while(true):
    spin_lock(lock_1)
    spin_trylock(lock_2)
    if not have both locks:
        spin_unlock(lock_1)
        continue
    break
```

Program_B:

```
while(true):
    spin_lock(lock_2)
    spin_trylock(lock_1)
    if not have both locks:
        spin_unlock(lock_2)
        continue
    break
```

PA	PB
L1	L2

Livelock!

Program_A:

```
while(true):
    spin_lock(lock_1)
    spin_trylock(lock_2)
    if not have both locks:
        spin_unlock(lock_1)
        continue
    break
```

Program_B:

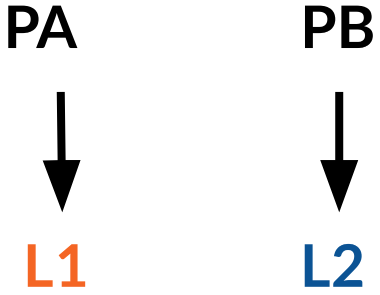
```
while(true):
    spin_lock(lock_2)
    spin_trylock(lock_1)
    if not have both locks:
        spin_unlock(lock_2)
        continue
    break
```

PA	PB
L1	L2

Livelock!

```
Program_A:

while(true):
    spin_lock(lock_1)
    spin_trylock(lock_2)
    if not have both locks:
        spin_unlock(lock_1)
        continue
    break
```

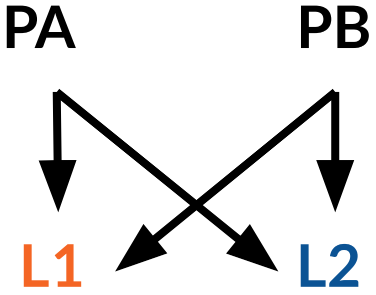
```
Program_B:

while(true):
    spin_lock(lock_2)
    spin_trylock(lock_1)
    if not have both locks:
        spin_unlock(lock_2)
        continue
    break
```



Livelock!

```
Program_A:

while(true):
    spin_lock(lock_1)
    spin_trylock(lock_2)
    if not have both locks:
        spin_unlock(lock_1)
        continue
    break
```

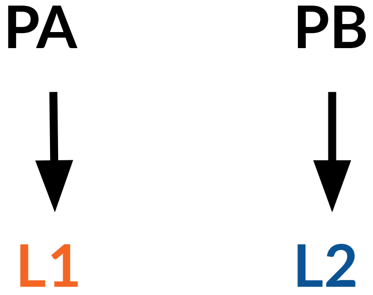
```
Program_B:

while(true):
    spin_lock(lock_2)
    spin_trylock(lock_1)
    if not have both locks:
        spin_unlock(lock_2)
        continue
    break
```



Livelock!

```
Program_A:

while(true):
    spin_lock(lock_1)
    spin_trylock(lock_2)
    if not have both locks:
        spin_unlock(lock_1)
        continue
    break
```

```
Program_B:

while(true):
    spin_lock(lock_2)
    spin_trylock(lock_1)
    if not have both locks:
        spin_unlock(lock_2)
        continue
    break
```



PA

PB

L1

L2

Livelock!

```
Program_A:

while(true):
    spin_lock(lock_1)
    spin_trylock(lock_2)
    if not have both locks:
        spin_unlock(lock_1)
        continue
    break
```

```
Program_B:

while(true):
    spin_lock(lock_2)
    spin_trylock(lock_1)
    if not have both locks:
        spin_unlock(lock_2)
        continue
    break
```

PA	PB
L1	L2

Livelock!

Program_A:

```
while(true):
    spin_lock(lock_1)
    spin_trylock(lock_2)
    if not have both locks:
        spin_unlock(lock_1)
        continue
    break
```

Program_B:

```
while(true):
    spin_lock(lock_2)
    spin_trylock(lock_1)
    if not have both locks:
        spin_unlock(lock_2)
        continue
    break
```

PA	PB
L1	L2

How to solve livelock?

- We can solve this problem by acquiring locks in the same order.
- We must release the locks in the reverse order they were acquired.

How to solve livelock?

Program_A:

```
while(true):
    spin_lock(lock_1)
    spin_trylock(lock_2)
    if not have both locks:
        spin_unlock(lock_1)
        continue
    break
```

Program_B:

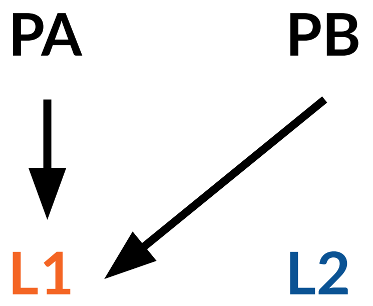
```
while(true):
    spin_lock(lock_1)
    spin_trylock(lock_2)
    if not have both locks:
        spin_unlock(lock_1)
        continue
    break
```

PA	PB
L1	L2

How to solve livelock?

```
Program_A:

while(true):
    spin_lock(lock_1)
    spin_trylock(lock_2)
    if not have both locks:
        spin_unlock(lock_1)
        continue
    break
```

```
Program_B:

while(true):
    spin_lock(lock_1)
    spin_trylock(lock_2)
    if not have both locks:
        spin_unlock(lock_1)
        continue
    break
```



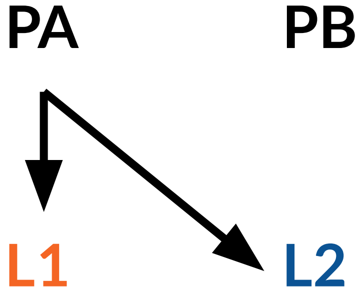
How to solve livelock?

Program_A:

```
while(true):
    spin_lock(lock_1)
    spin_trylock(lock_2)
    if not have both locks:
        spin_unlock(lock_1)
        continue
    break
```

Program_B:

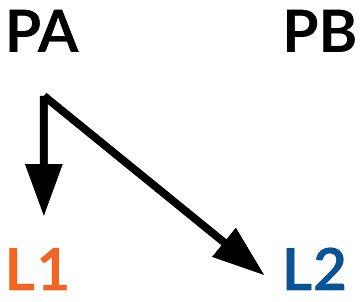
```
while(true):
    spin_lock(lock_1)
    spin_trylock(lock_2)
    if not have both locks:
        spin_unlock(lock_1)
        continue
    break
```



How to solve livelock?

```
Program_A:

while(true):
    spin_lock(lock_1)
    spin_trylock(lock_2)
    if not have both locks:
        spin_unlock(lock_1)
        continue
    break
```

```
Program_B:

while(true):
    spin_lock(lock_1)
    spin_trylock(lock_2)
    if not have both locks:
        spin_unlock(lock_1)
        continue
    break
```



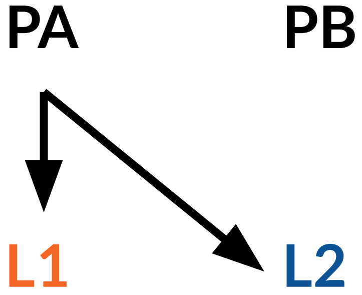
How to solve livelock?

Program_A:

```
while(true):
    spin_lock(lock_1)
    spin_trylock(lock_2)
    if not have both locks:
        spin_unlock(lock_1)
        continue
    break
```

Program_B:

```
while(true):
    spin_lock(lock_1)
    spin_trylock(lock_2)
    if not have both locks:
        spin_unlock(lock_1)
        continue
    break
```



How to solve livelock?

Program_A:

```
while(true):
    spin_lock(lock_1)
    spin_trylock(lock_2)
    if not have both locks:
        spin_unlock(lock_1)
        continue
    break
```

Program_B:

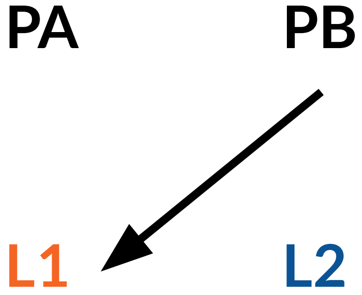
```
while(true):
    spin_lock(lock_1)
    spin_trylock(lock_2)
    if not have both locks:
        spin_unlock(lock_1)
        continue
    break
```

PA	PB
L1	L2

How to solve livelock?

```
Program_A:

while(true):
    spin_lock(lock_1)
    spin_trylock(lock_2)
    if not have both locks:
        spin_unlock(lock_1)
        continue
    break
```

```
Program_B:

while(true):
    spin_lock(lock_1)
    spin_trylock(lock_2)
    if not have both locks:
        spin_unlock(lock_1)
        continue
    break
```



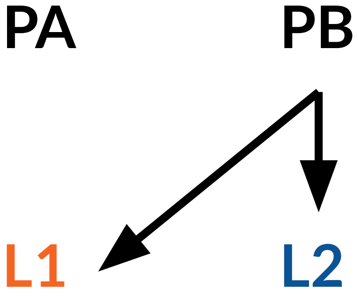
How to solve livelock?

Program_A:

```
while(true):
    spin_lock(lock_1)
    spin_trylock(lock_2)
    if not have both locks:
        spin_unlock(lock_1)
        continue
    break
```

Program_B:

```
while(true):
    spin_lock(lock_1)
    spin_trylock(lock_2)
    if not have both locks:
        spin_unlock(lock_1)
        continue
    break
```



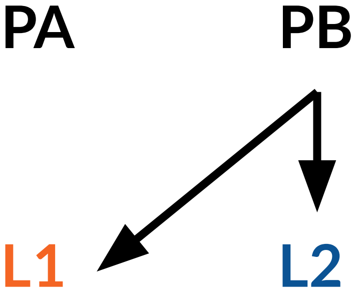
How to solve livelock?

Program_A:

```
while(true):
    spin_lock(lock_1)
    spin_trylock(lock_2)
    if not have both locks:
        spin_unlock(lock_1)
        continue
    break
```

Program_B:

```
while(true):
    spin_lock(lock_1)
    spin_trylock(lock_2)
    if not have both locks:
        spin_unlock(lock_1)
        continue
    break
```



PA

PB

L1

L2

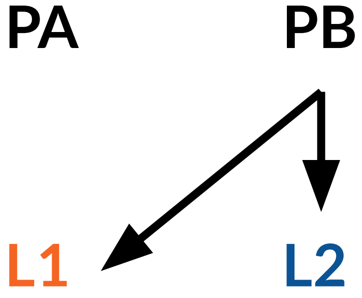
How to solve livelock?

Program_A:

```
while(true):
    spin_lock(lock_1)
    spin_trylock(lock_2)
    if not have both locks:
        spin_unlock(lock_1)
        continue
    break
```

Program_B:

```
while(true):
    spin_lock(lock_1)
    spin_trylock(lock_2)
    if not have both locks:
        spin_unlock(lock_1)
        continue
    break
```



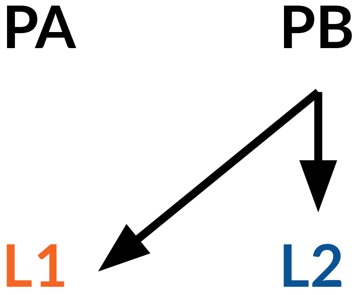
How to solve livelock?

Program_A:

```
while(true):
    spin_lock(lock_1)
    spin_trylock(lock_2)
    if not have both locks:
        spin_unlock(lock_1)
        continue
    break
```

Program_B:

```
while(true):
    spin_lock(lock_1)
    spin_trylock(lock_2)
    if not have both locks:
        spin_unlock(lock_1)
        continue
    break
```



Synchronization Example

- There is another synchronization method other than the ones taught in class called a barrier. Barriers make sure that all threads stop at a `barrier_wait` point before continuing. Fill in the following struct and writing the following functions. Assume `NUM_THREADS` threads exist
  - `typedef struct{spinlock_t lock;`
    `…more variables…}barrier_t;`
  - `void barrier_init(barrier_t *b){…code…}`
  - `void barrier_wait(barrier_t *b){…code…}`

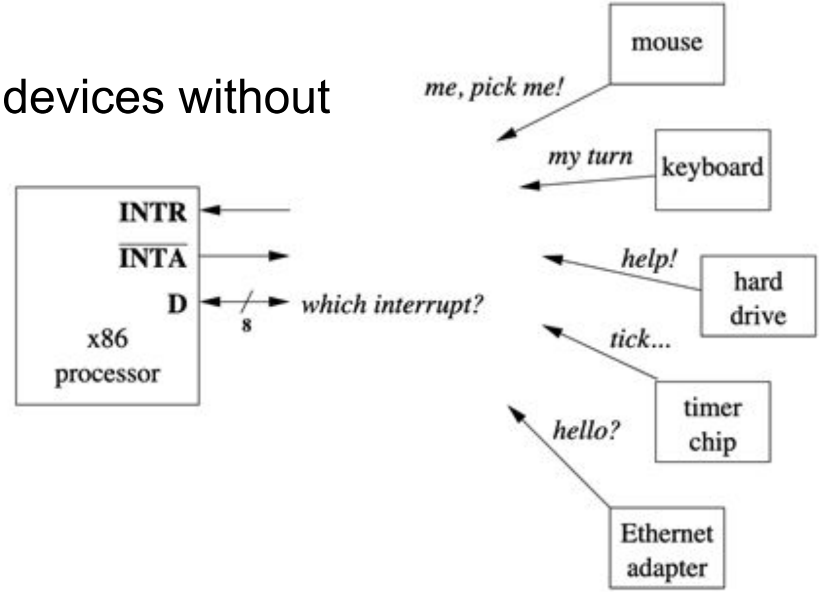
```
typedef struct{spinlock_t thread_count_lock;
    volatile int threads_joined;} barrier_t;
void barrier_init(barrier_t *b){
    if(b == 0) return;
    b->thread_count_lock = SPIN_LOCK_UNLOCKED;
    b->threads_joined = 0;
}
void barrier_wait(barrier_t *b){
    if(b == 0) return;
    spin_lock(&(b->thread_count_lock));
    b->threads_joined++;
    spin_unlock(&(b->thread_count_lock));
    while(b->threads_joined != NUM_THREADS);
    return
}
```

Questions?

PIC (Programmable Interrupt Controller)

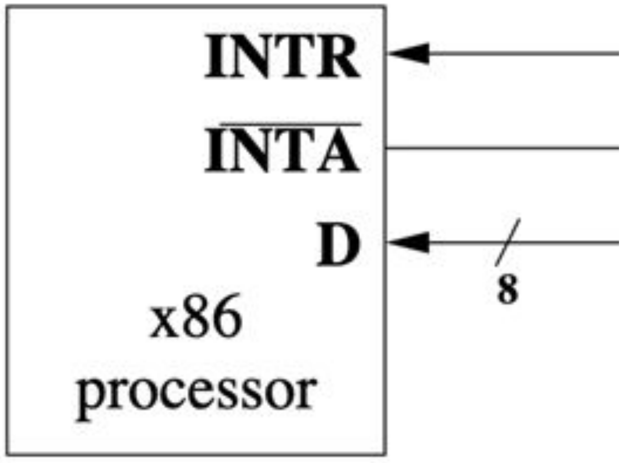
Why do we need a PIC?

- We want a way to interact with computers
  - Device I/O
- PIC offers a way to manage multiple devices without adding processor side complexity
  - Arbitrates b/w device requests
  - Enforces priority



PIC Terminology

- The CPU and PIC communicate via *ports*
    - 0x20, 0x21
- INTR - signal telling the CPU an int is generated
- INTA’ - signal telling the PIC the CPU received the signal
- D - a bus that can send int number to the CPU



Data and Address Bus (DAB)

- Address Bus: 16 bits that specifies what port we are reading from
  - Examples: 0x20, 0x21

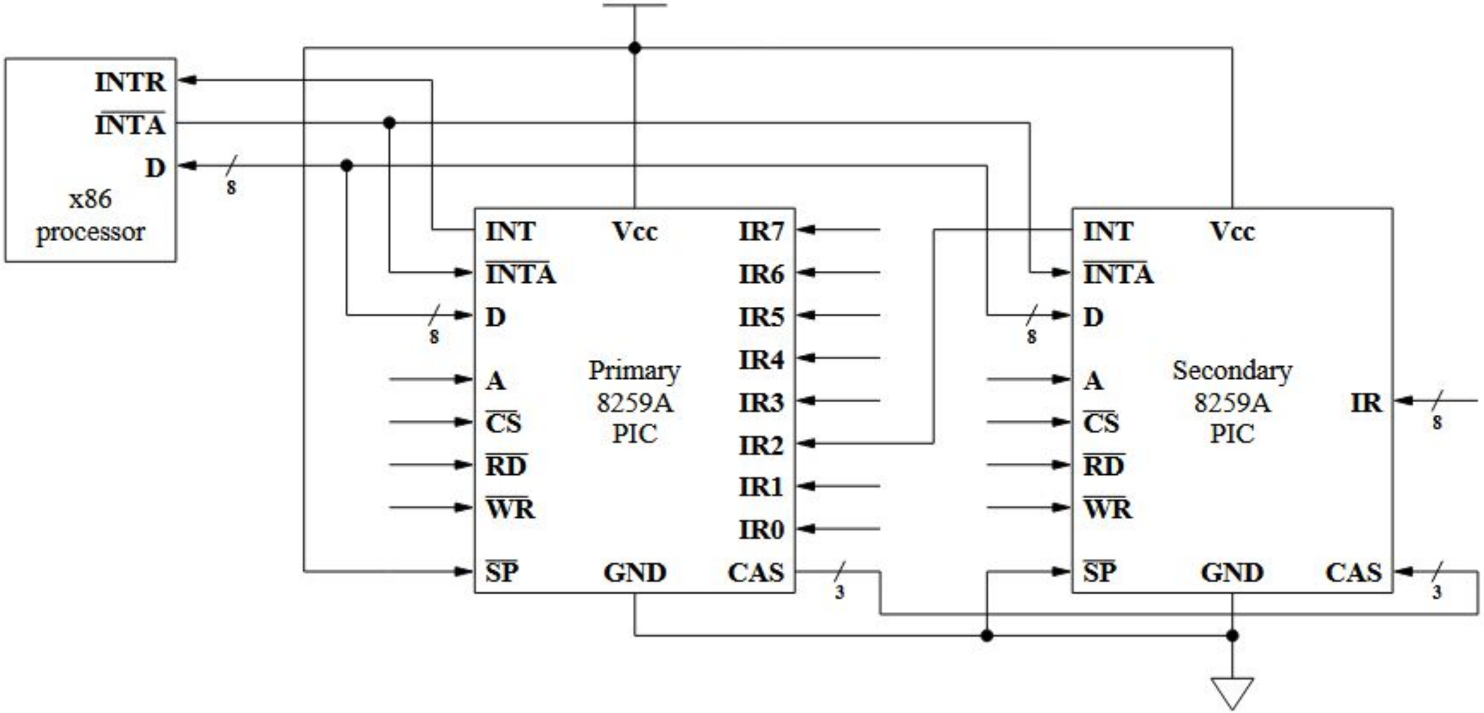
- Data Bus: 8 bits of data coming from the port specified in the address bus
  - Examples: the DATA coming out of port 0x20, 0x2

Main PIC I/O

- A: Address_bus[0] -> to differentiate between command/ data
- CS’: Address_bus[15:1] = ? 0x10
- RD’: processor will read from the port (when processor uses IN)
- WR’: processor will write to the port (when processor uses OUT)

What if we need more than 8 devices connected at a time?

Cascading PICs



More I/O

- SP’: Used to identify whether the PIC is the primary or a secondary PIC
- CAS: Used by the primary PIC to identify which secondary PIC should write to the data line

PIC Example

- Three 8259A PICs are cascaded together, with a secondary X occupying IR0 on primary and secondary Y occupying IR4 on primary. Assuming that the standard priority scheme is used, show the overall priority scheme of interrupts. Use P0 through P7 for the primary PIC and X0 though X7 and Y0 thought Y7 for the two secondary PICs.

PIC Example

- Three 8259A PICs are cascaded together, with a secondary X occupying IR0 on primary and secondary Y occupying IR4 on primary. Assuming that the standard priority scheme is used, show the overall priority scheme of interrupts. Use P0 through P7 for the primary PIC and X0 though X7 and Y0 thought Y7 for the two secondary PICs.

X0 … X7, P1, P2, P3, Y0 … Y7, P5, P6, P7

PIC Example

- Draw/Describe the necessary glue logic to connect the address and chip select ports of the PIC to the 16-bit address bus of a processor such that the PIC occupies ports 0x100 and 0x101.

PIC Example

- Draw/Describe the necessary glue logic to connect the address and chip select ports of the PIC to the 16-bit address bus of a processor such that the PIC occupies ports 0x100 and 0x101.

A = Address[0]
CS’ = if Address[15:1] is not 0x80

Questions?