

# Architecture, Design and Implementation of Embedded Systems for Real-Time Applications

## CpE-450 Spring 05

### Class 8

Bruce McNair

[bmcnair@stevens.edu](mailto:bmcnair@stevens.edu)


# Loop timing with interrupts

```
main()
{
    start_timer(500);
    enable_interrupts();
    while(1)
    {
        perform_background_processing();
        halt_processor(WAKE_ON_INT);
    }
}

void IO_ISR(void)
{
    disable_interrupts();
    do_IO();
    enable_interrupts();
}

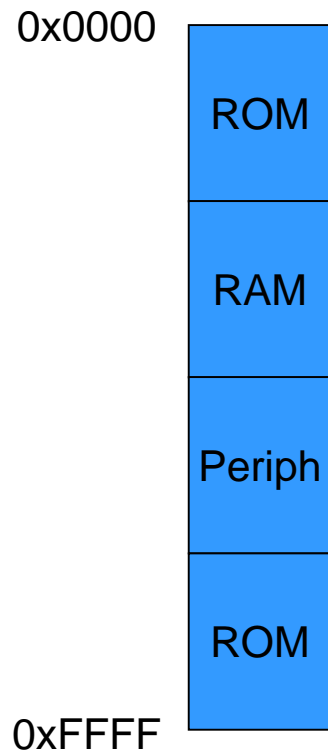
void blink_ISR (void)
{
    restart_timer(500);
    disable_interrupts();
    TOGGLE_LED();
    do_other_stuff();
    enable_interrupts();
}
```

No unnecessary functions  
Minimal power consumption  
Wastes no real-time



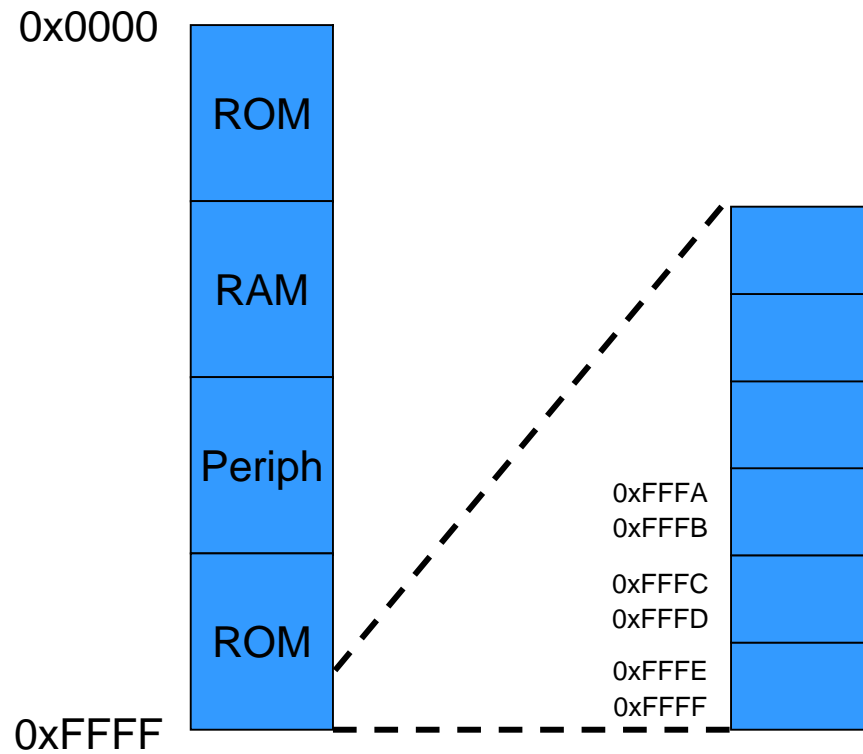
# Vectored Interrupts

- Example memory map:



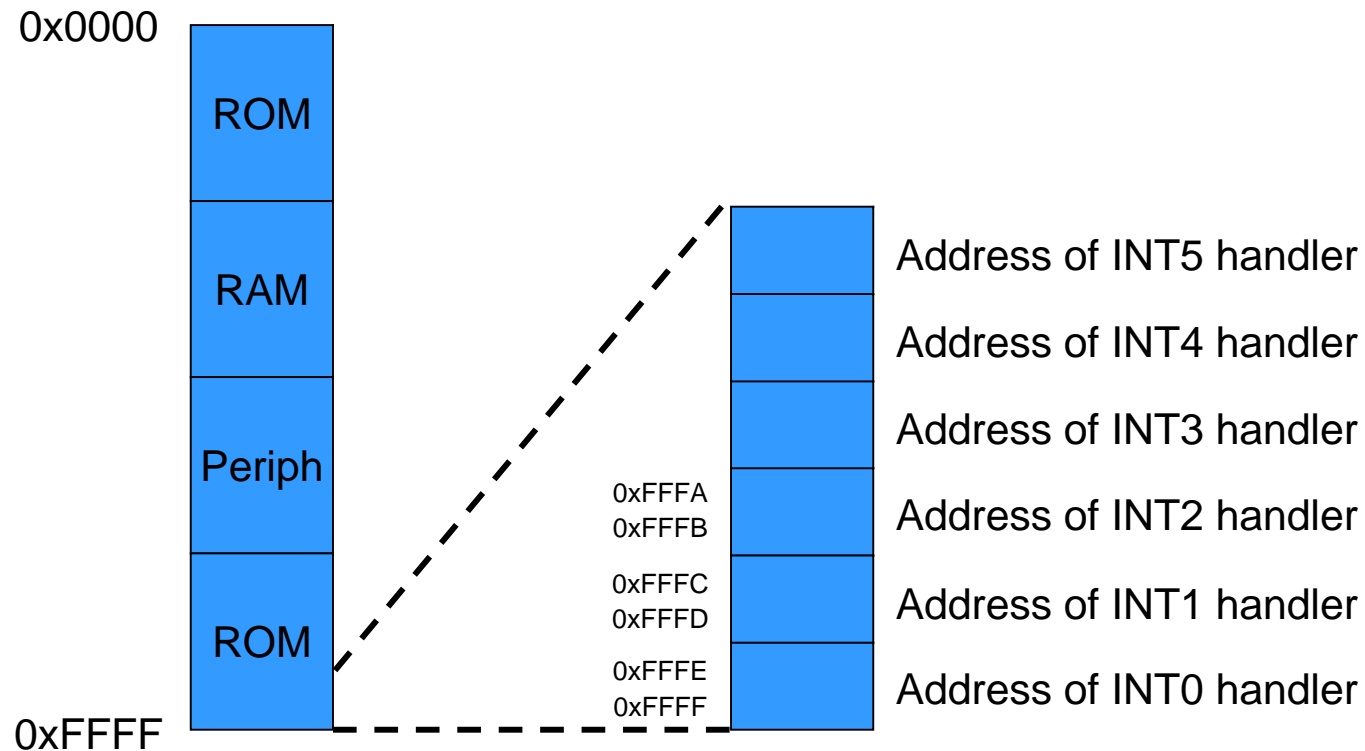
# Vectored Interrupts

- Example memory map:



# Vectored Interrupts

- Example memory map:



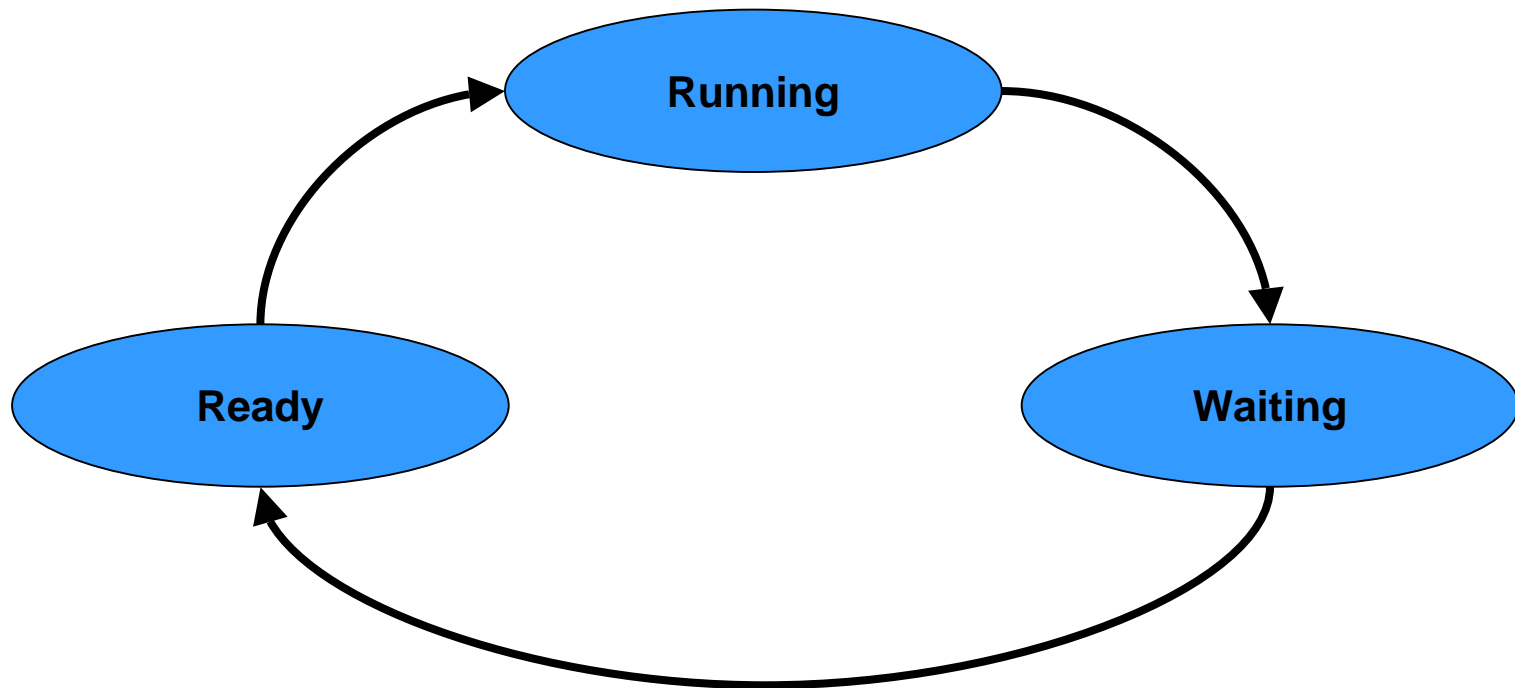
# Interrupt Processing

- Example pseudo-microcode

```
void microcomputer_microcode(void)
{
    while(1)
    {
        while(!interrupt())
        {
            fetch_instruction(addr);
            execute_instruction();
            addr++;
        }
        push(addr);
        addr := handle_interrupt(int_level)
    }
}

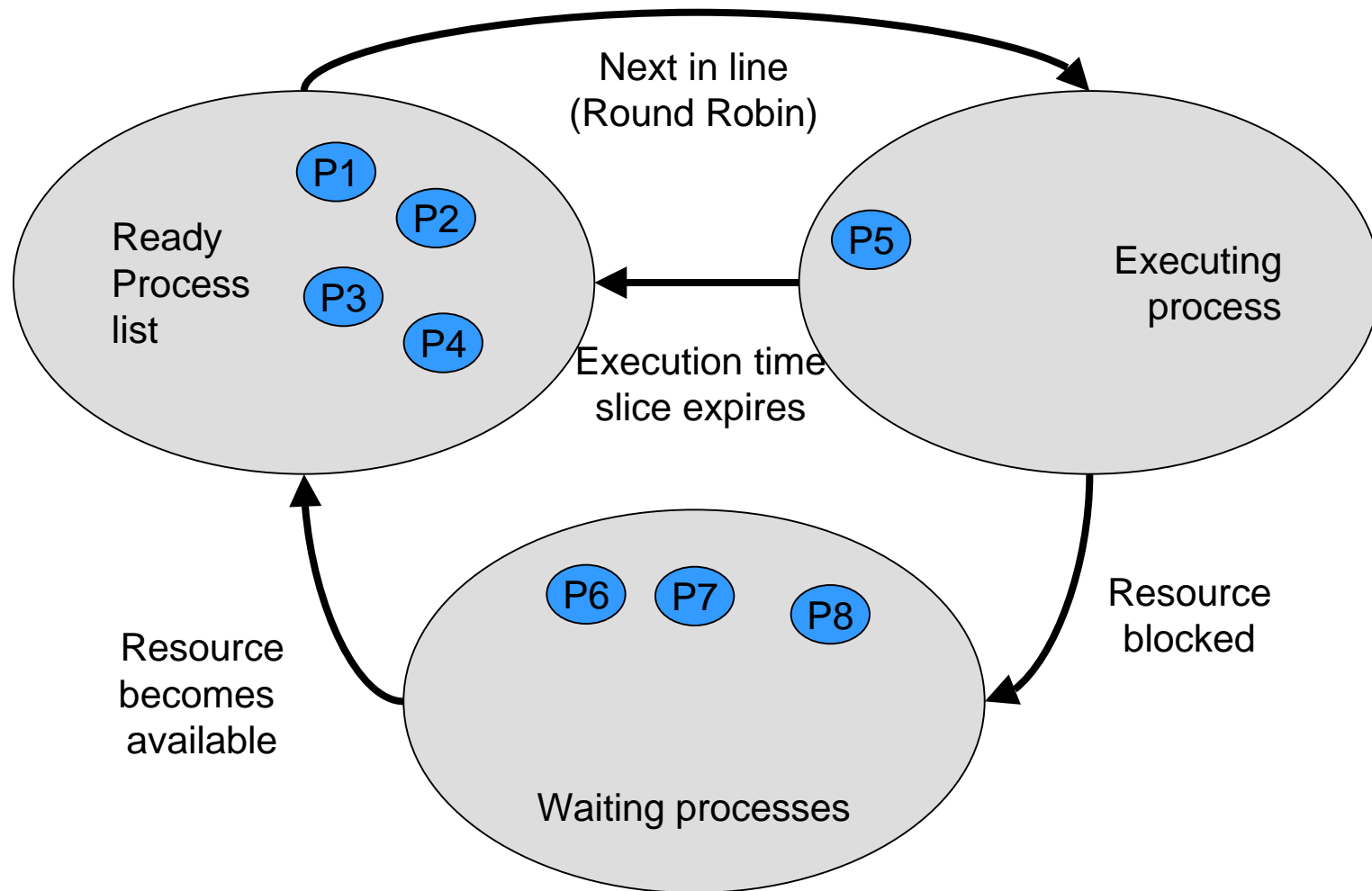
unsigned int *handle_interrupt(unsigned int int_level)
{
    unsigned int_vec;
i := get_interrupt_level();
    int_vec := 0xFFFF - 2*int_level - 1;
    return( (unsigned int *)int_vec );
}
```

# Process Scheduling



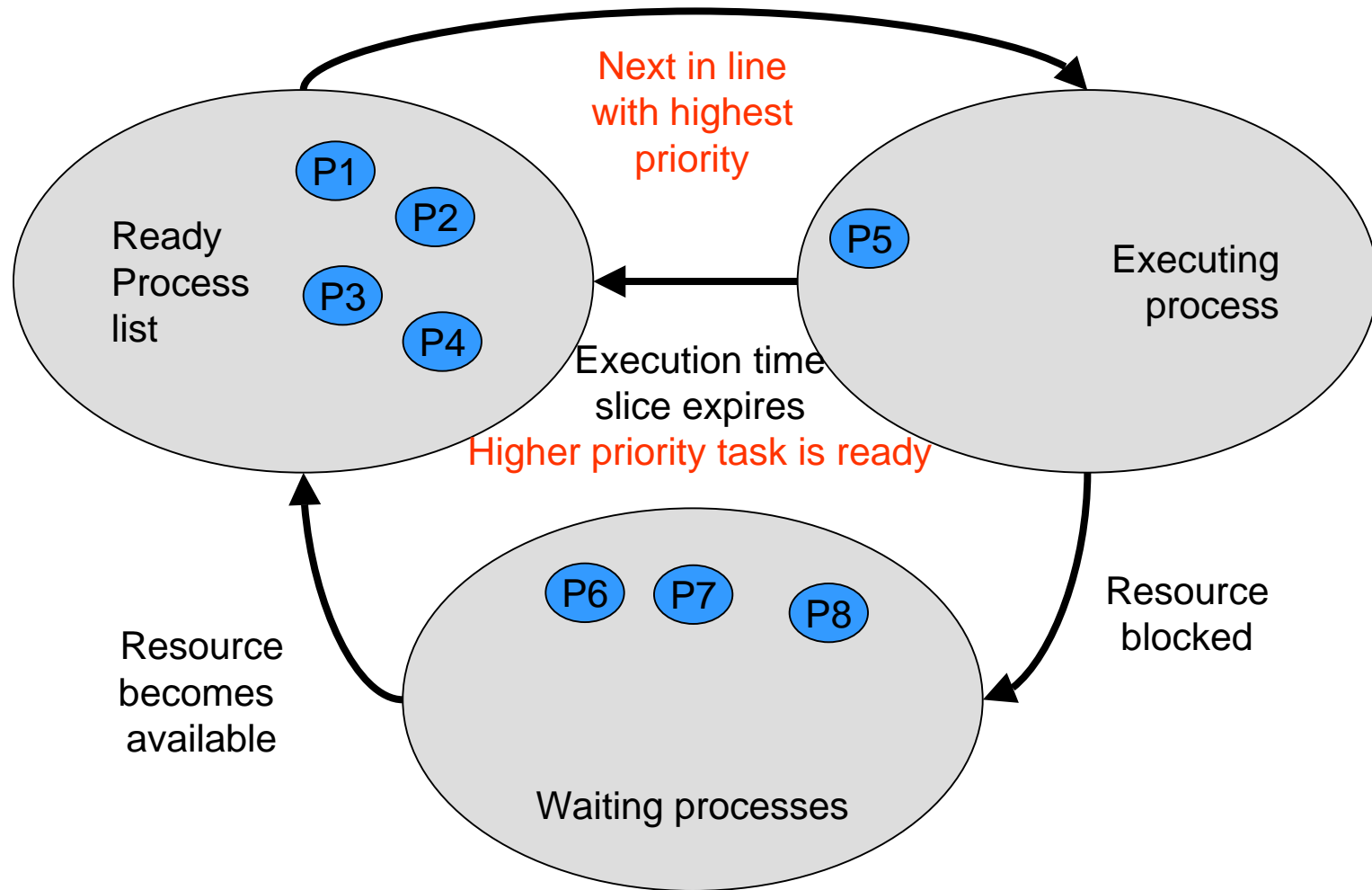
# Choosing a process to run

- Standard OS:



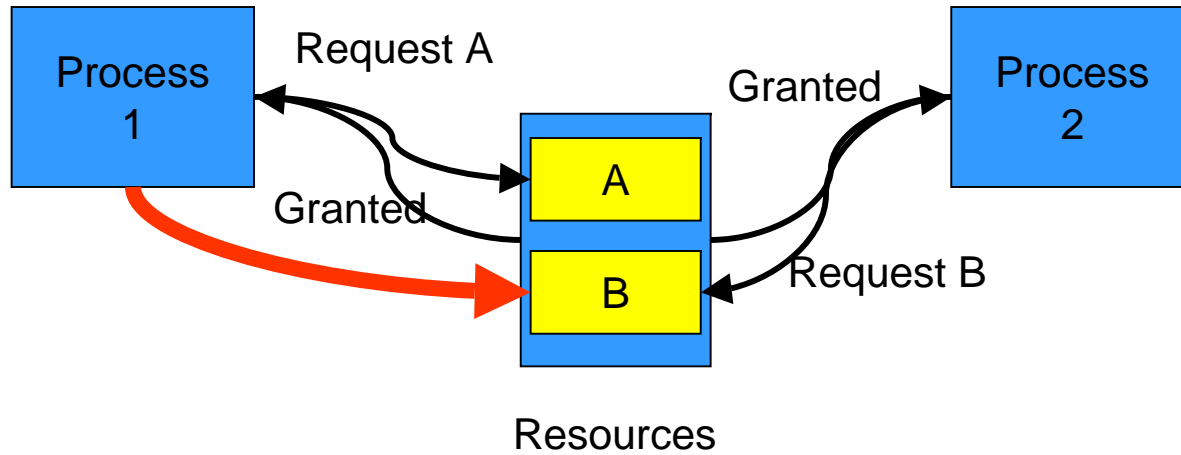
# Choosing a process to run

- Real-time OS:



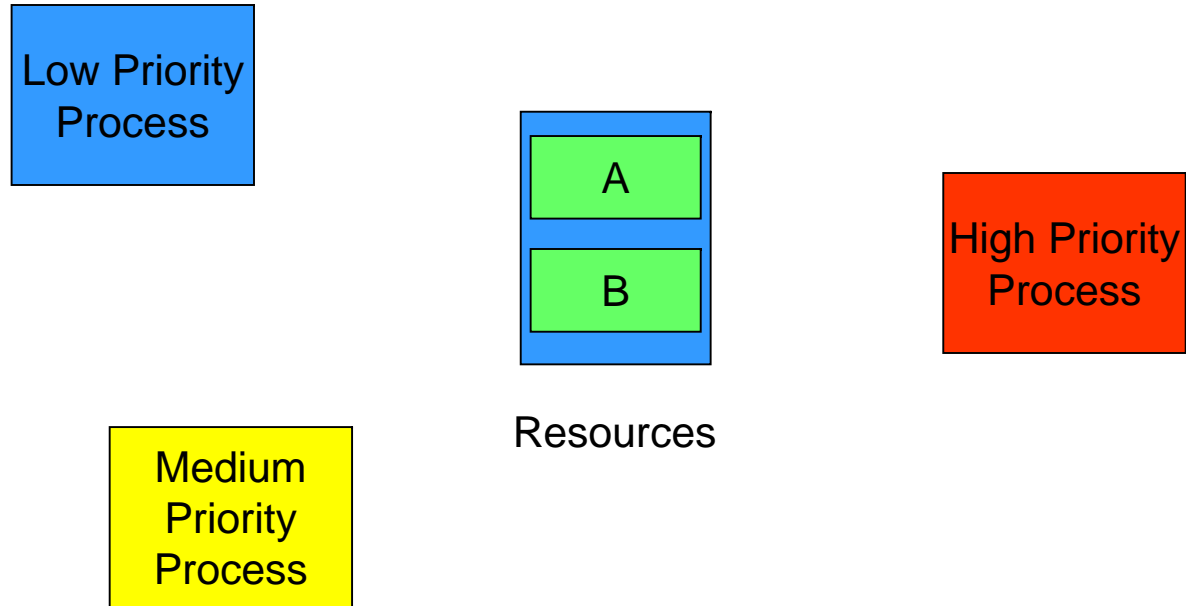


# Shared Resources and Process Contention

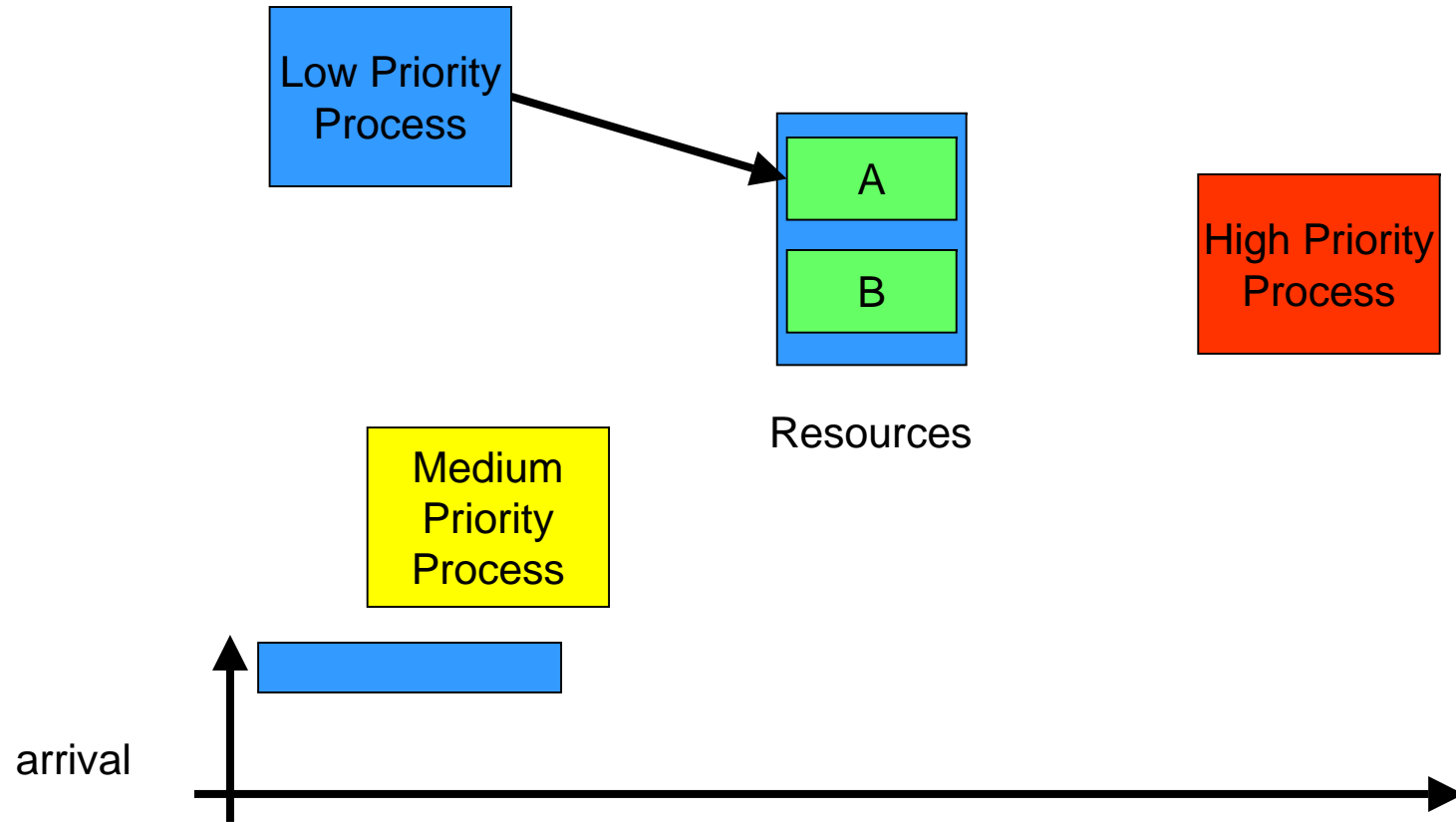


- Shared resources and contending processes create the potential for deadlock

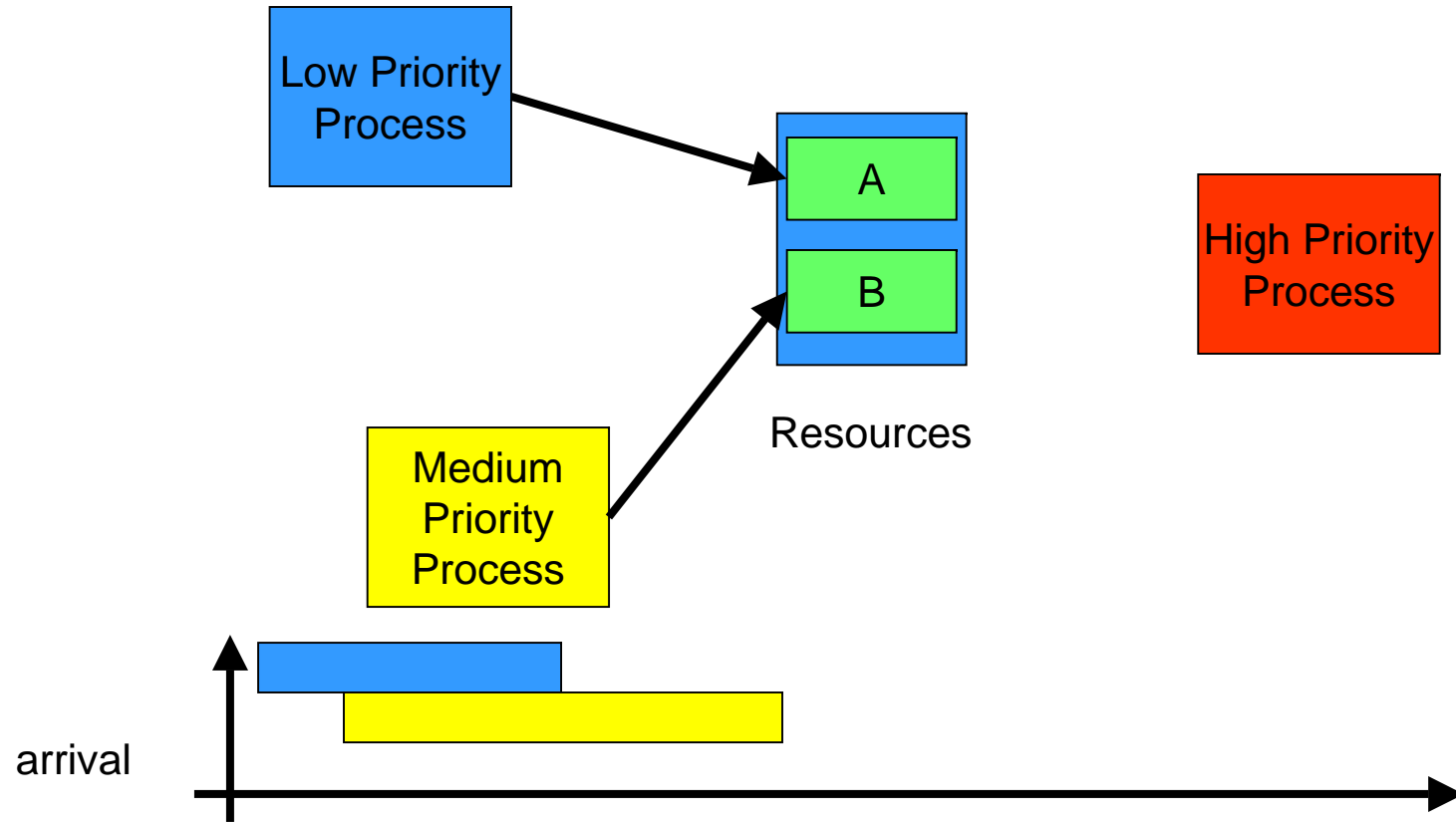
# Task Priority Inversion



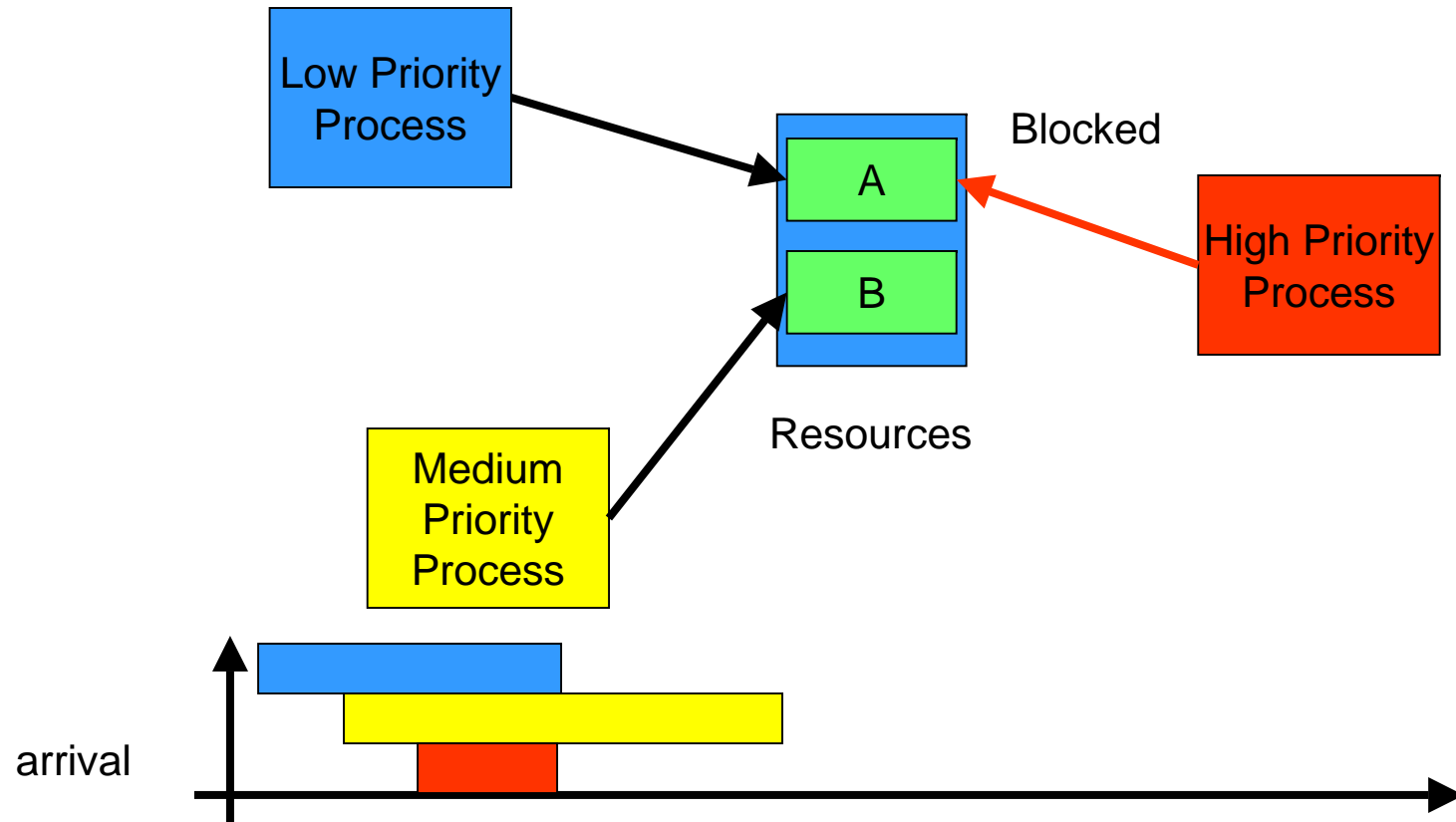
# Task Priority Inversion



# Task Priority Inversion



# Task Priority Inversion



# Task Priority Inversion

