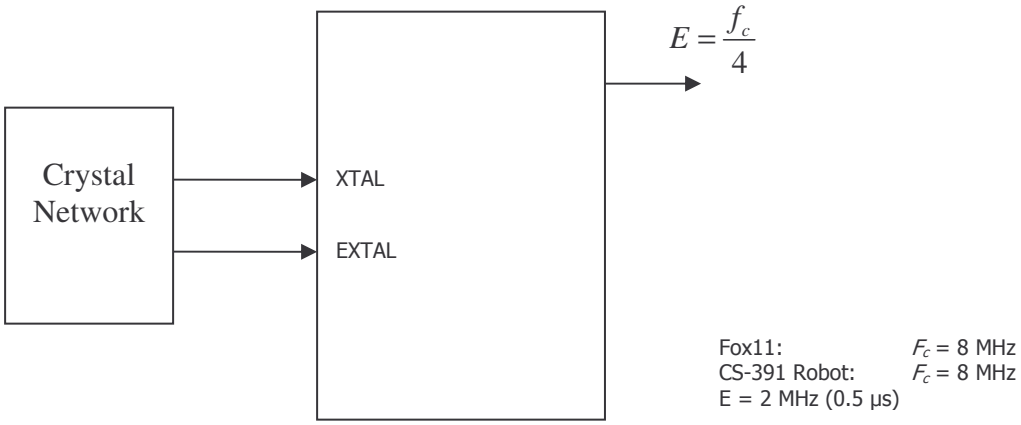
Key parts of the OS:

Bootstrap
Kernel
Device drivers

Everything on a processor is controlled by time.



A segment of code that is executing is a process.
The kernel is a process. It happens to be a special process. It is always resident in memory.

We have to switch processes fast enough so that users do not notice delay.

This is a context switch and provides the basic kernel functionality. This is created by writing a timer whose ISR is the kernel.

Quantum Time

Amount of time a process is allowed to run.

At the end of the quantum time, an interrupt occurs to context switch.

Preemptive Multitasking

Switches **process contexts** every periodic quantum time. In general, it is done with a timer device that interrupts the CPU.

Context switch

Given: quantum time = 10 ms and an ideal kernel, how many context switches can occur in 1 second?
100
If you have 200 processes, will the context switches be noticeable? Probably not, as the majority of those processes will be sleeping.

Goal of first lab: to set up a quantum time interrupt so that we can have the kernel ISR move in and function as it should to switch processors.

MC68HC11 Timer Device

1. Real Time Interrupt – RTI
    a. Periodic interrupt at programmable periods
2. Output Compare
3. Input Capture
4. Pulse Accumulation

Device Programming

Initialize control registers
Enable interrupts

Interrupt Occurs

1. CPU stops running a program
2. Stores A, B, X, Y, PC, CCR, SP onto the stack – machine state save
3. Discovers the device that interrupted and the type of interrupt
4. PC = mem[interrupt index] (starting address of ISR – contained in interrupt vector table)

```
LDAA #0x7E
STAA 0x00EB
LDX  #RTIISR
STX  0x00EC

RTIISR:     LDAB
            ...
            RTI

ORG  0x00EB
JMP  RTIISR
```

ORG is a command to the assembler that tells the assembler at that place in memory to start placing instructions.