Karan Kumar 11114021

BULDING MICRO VERSION OPERATING SYSTEM

To create a kernel, we need a compiler that translates C to assembly as well as a toolchain for assembly, linking, and other useful functions. However these programs produce programs for local GNU/Linux system, not the target system. We'll therefore need to set up a cross compiler that produces binaries for another operating system, which will be the first task.

Creating a Cross-Compiler Toolchain

We are using GNU's Compiler Collection (gcc) version 4.7.2 and GNU Binutils version 2.23.1 for this purpose. GCC provides the compiler infrastructure and binutils provides the infrastructure for assembly, linking, disassembly, and other useful programs.

Building the Cross Binutils

In this section we build a binutils for the x86 CPU.

Building the Cross Compiler

In this section, we will build the cross compiler that will translate operating system to machine code.

Booting the Operating System

To start operating system, an existing piece of software will needed to load it.

IMPLEMENTATION OF KERNAL

```
void kmain()
{
      // TODO: Kernel here.
}
```

Setting up the Linker Script

The linker script tells linker how to construct the final kernel image. The executable kernel consists of a number of segments tell where to load code from the file and where.

Writing a Makefile

Writing a Makefile simplifies the building of the operating system as well as documenting how it is done.

Finishing up

We are using Virtual machine to test the Kernal.