Week 2

Installed Required Tools: Feb 8, 2019 5:00 pm

I installed the required tools to compile run JOS on Ubuntu. For more info refer to the section below:

Tools Required

For now we will need following tools:

1. **QEMU:** It is an open source virtual machine monitor. We will be using this to run JOS.
2. **Compiler Toolchain:** We would also need compiler toolchain. A compiler toolchain consists of various different programs including C compiler, assemblers and linker. We need it to convert the code written in assembly and C to machine language.

Check if you have tools installed:

1. Execute `objdump -i` and the second line should include `elf32-i386` .
2. Try executing `gcc -v` and it should run successfully.

Installing Required tools:

If the commands above do not give the desired output then you can install them using following commands:

Linux:

```
sudo apt-get install -y build-essential gdb
sudo apt-get install gcc-multilib
sudo apt-get install qemu
```

Mac OS X:

Note: You need to have homebrew installed for this to work.

```
xcode-select --install
brew install $(brew deps qemu)
brew install qemu
```

Writing minimal programs for languages to be used: Feb 9, 2019 1:00 pm

Assembly: NASM

Good knowledge of assembly is required to complete Lab 1 for this project as it focuses on getting the JOS to boot by writing code for boot loader.

Assembly: Hello World

This is basic assembly code that just prints “Hello World!” to the screen (See reference: Assembly Tutorial).

Commands to assemble, link and run this file:

1. cd into examples folder.
2. Execute `mkdir bin`
3. Assemble the program by executing `nasm -f elf helloworld.asm -o ./bin/helloworld.o`
4. Link the object file to create executable execute `ld -m elf_i386 -s -o helloworld helloworld.o`
5. Execute the executable by running `./bin/helloworld`

Barebone Bootloader

I continue to work on the bootloader code that we wrote in our second class for COMP 340 and have developed this barebone bootloader that just prints hello world on boot.

Problem Faced: Understanding the code was bit of a challenge as it was bit different from what we did cover in COMP 256.

Solution: OS Dev and Stack Overflow were a great reasource that really helped me understand how boot loader code worked.

Commands to create binary and boot the computer off that binary:

1. cd into examples folder.
2. Execute `mkdir bin`
3. Assemble the program by executing `nasm -f bin bootloader.asm -o ./bin/bootloader.bin`
4. Fire a VM that uses our bootloader using the following command `qemu-system-x86_64 ./bin/bootloader.bin` .

C Language

Majority of JOS project is written in C. So there is no doubt that C is of great significance for this project.

C: Hello World

This is a simple Hello World application. It just prints “Hello World!” to the screen and exits. If we are able to compile and run this project then we can say that our C compiler is up and running.

Commands to compile and run this project:

1. cd into examples folder.
2. In order to compile a C file we need to execute `gcc <fileToCompile.c> -o <outputFile.out>` . For this example program we can execute `gcc helloworld.c -o helloworld.out`
3. After above command executes succesfully we can run our compiled program by just typing in name of the output file i.e. in our case just execute `./outputFile.out`

References for Week 2

- Tools Used: https://pdos.csail.mit.edu/6.828/2018/tools.html
- Bootloader: https://wiki.osdev.org/Bootloader
- Assembly Tutorial: https://www.tutorialspoint.com/assembly_programming/index.htm
- What does double dollar sign mean in x86: https://stackoverflow.com/questions/46726434/what-does-double-dollar-sign-mean-in-x86-assembly-nasm