CS340400 Compiler Design
Qemu Simulation Guide

Outline

1. Generate Executable
    1. codegen.S
    2. Compile Executable
2. RISCV-V Qemu Simulator
    1. Qemu Usage

Generate Executable

1.1 codegen.s

- The rules are totally the same as those codegening for Andes Corvette-F1-N25, including but not limited to the following items
  - Same set of testcases
  - Implement **delay**, **digitalwrite**
  - **.global codegen**
  - …

1.2 Compile Executable

- TAs provide a tweaked version of the assembly sample project, which includes:
  - **main.c** : The main program
  - **codegen.s** : The same one as in the **assembly** project

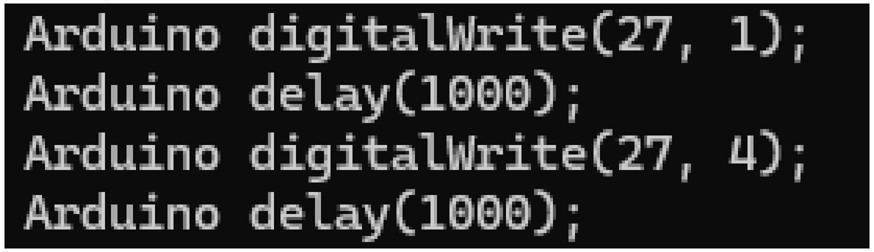
1.2 Compile Executable(cont.)

- To compile your **codegen.S** into an executable, use **riscv32-unknown-elf-gcc**
  - E.g. $ **riscv32-unknown-elf-gcc -o sample_prog main.c codegen.S** in the assembly folder
    - The above command does the following:
      - Compile **main.c**
      - Assemble **codegen.S**
      - Link them together to produce **sample_prog**

RISCV-V Qemu Simulator

2.1 Qemu Usage

- Suppose we have our compiled **sample_prog** , to execute it, run:
  - **$ qemu-riscv32 sample_prog**
  - You should have a correct invocation log of **delay** and **digitalWrite** as output
    - Output:



```
Arduino digitalWrite(27, 1);
Arduino delay(1000);
Arduino digitalWrite(27, 4);
Arduino delay(1000);
```