

CS3160: RISC-V Assembly Programming

1 RISC-V Programming Guide

1.1 Files and Folders (conventions followed)

1. We will use the following conventions:

***.s** The .s (and not .S) file extension is used for the RISC-V source code (assembly code).

***.r5o** The file extension .r5o is used to indicate a RISC-V ELF.

***.r5o.dump** Output of objdump on a RISC-V ELF.

***.iss** Output of the Spike run.

2. The folder structure (of our pr5) so far is as follows:

pr5/ Top-level or Project Root folder. All the code and its associated files will reside inside this folder.

pr5/programs Folder where all the RISC-V assembly programs we write will reside.

pr5/programs/asms/ Folder that contains all the RISC-V assembly programs we will write (as practice and as assignment). All files should end with the extension .s (and not .S).

pr5/programs/c_tests/ Folder that contains all C programs. Write any C program that you want to practice and learn how C programs are converted to RISC-V machine instructions.

pr5/programs/custom/ Folder that contains additional files to generate RISC-V ELF (linker scripts, headers, etc).

pr5/programs/bins Auto-generated. Contains all the RISC-V executables (.r5o) grouped under subfolders **c_tests** and **asms**.

pr5/programs/dumps Auto-generated. Contains objdumps (only .text and .data) (again grouped under subfolders **c_tests** and **asms**).

pr5/programs/runs Auto-generated. Contains the output produced when executables are run on Spike (again grouped under subfolders **c_tests** and **asms**).

3. The files (in our pr5) so far are:

pr5/programs/Makefile Contains commands to compile assembly programs, generate objdump, and run executables on spike.

pr5/programs/asms/endlessloop.s A simple loop that does not terminate (used to halt system).

pr5/programs/asms/datavars.s Declares global variables myvar1, myvar2 and array myarr initialized with sample values. Main loads addresses into registers.

1.2 Compiling (or assembling) programs

1. Write your assembly programs in pr5/programs/asms. File extension must be .s.
2. Ensure riscv-none-elf-gcc is in your \$PATH.
3. Compile with:

```
cd pr5/programs  
make
```

This creates bins/ and dumps/ folders.

1.3 Running RISC-V programs

1. Ensure spike is in your \$PATH.
2. Run all programs:

```
cd pr5/programs  
make run_all
```

This creates *.iss files showing instruction sequence, memory access, etc. You can also use run_asms or run_c_tests instead of run_all to run only the assembly code or C programs, respectively.

1.4 Additional Help and Reading Materials

1. Chapter 2 of Computer Organization and Design – RISC-V Edition (RV64I ISA, but we use RV32I).
2. Use riscv-none-elf-gcc -O0 -S to generate assembly.
3. Use riscv-none-elf-objdump -d to inspect executables.

2 Practice Problems

1. Write a RISC-V assembly program that initializes register x10 to 0x1234abcd.
2. Write a program to find the maximum of three numbers (in x1, x2, x3).
3. Write a program that adds x1 + x2 and stores result in global variable sum.
4. Write a program that initializes a global array arr[100] with 0xc001 using a loop.

3 Lab Assignment 1 (Graded)

1. Write a program to count the number of positive ($>= 0$), even numbers in array 1 (size $n > 0$). Place result in $x10$. Template: `asms/1-even.s`.
2. Write a program to check if a number is prime. Place 1 in $x10$ if prime, else -1. Template: `asms/2-prime.s`.
3. Write a program to sort an array in descending order (in-place). Template: `asms/3-descending.s`.
4. Write a program to compute histogram of marks obtained by n students. Store result in count array. Template: `asms/4-histogram.s`.
5. Write a RISC-V assembly program that defines the function:

```
int matmul(int *A, int *B, int *C);
```

Here A, B and C are 3×3 matrices, and $C = A \times B$. The function returns 0 if at least one of the elements is 0, and 1 otherwise. Write the program in a file named `asms/5-matmul.s`.

You can start with the template code given previously, but make sure that `matmul()` is a separate function that is called from `main()` (and adheres to the RISC-V calling conventions).

6. Write a recursive code to compute the factorial of a number. Write the program in a file named `asms/6-fact.s`.

You can start with the template code given previously, but make sure that `fact()` is a separate function that is called from `main()` (and adheres to the RISC-V calling conventions).