RISC-V Assembler

Submitted to: Dr. T.V. Kalyan

PREREQUISITES:

- Python 3.6 or above
- A text editor
- Terminal (Ubuntu/Linux) or Command Prompt (Windows)

Our ASSEMBLER supports the following instructions:

R format - add, and, or, sll, slt, sra, srl, sub, xor, mul, div, rem
I format - addi, andi, ori, lb, ld, lh, lw, jalr
S format - sb, sw, sd, sh
SB format - beq, bne, bge, blt
U format - auipc, lui
UJ format - jal

Our ASSEMBLER supports 64-bit too.

It supports the following assembler directives:
.text, .data, .byte, .half, .word, .dword, .asciiz.

INSTRUCTIONS about what it supports and what not:

- If there exist any instruction other than those 31 specified above, the program would halt there and show error on the terminal window.

- The arguments of the instructions can be separated by commas(,) or white spaces( ).

- If the number of arguments given are less or more than expected, the program will halt with an error message.

- The format of labels expected is:
  <label_name>:

- The ld, lw, lh, lb instruction also takes 3 arguments.
  Eg : lw x3,0(x2)
  It does not accept the 2 argument input format as they are pseudo instruction . For that we have to use 2 other basic instructions aupic and lw.
  If user is interested in using lw x1,label type instruction, it can be easily done using two instructions:

  1. *auipc <destination_register> 0x10000*
  2. *lw <destination_register>, <calculate the offset accordingly>(<destination_register>)*

- We have assumed that the memory allocation of the data in .data part of code is starting with the location: 0x10000000

HOW TO RUN:-

- We have 4 test codes:--
  1. Fibonacci ---- *data.asm*
  2. Bubble sort ---- *data4.asm*
  3. Factorial ---- *data2.asm*
- For running the code All you have to do is:
  1. Compile and Run *start.c* to run the phase1 and phase2 codes.:
     $gcc start.c
     $./a.out
  2. It will show the following request:-
     Which File You wanna execute?
     press 1 for data.asm
     press 2 for data2.asm
     press 3 for data4.asm
  3. Now press 1 or 2 or 3 as per your convenience and get the outputs in out1.txt, coded.mc for phase1 and out2.txt for phase2.

Contributions:-

Sweety: add, and, or, sll, slt, sra, srl, sub, xor, mul, div, rem --Phase1
auipc, lui, jal --Phase2
Garima: addi, andi, ori, lb, ld, lh, lw, jalr --Phase1
addi, andi, ori, lb, ld, lh, lw, jalr--Phase2
Diksha: GUI Work, will be submitted with next submission.
Anamika: beq, bne, bge, blt, auipc, lui, jal--Phase1
beq, bne, bge, blt--Phase2
Ruchika: sb, sw, sd, sh--Phase1
sb, sw, sd, sh --Phase2
Contributed some in GUI too.

Submitted by:

Garima Soni	2018csb1089
Anamika Sharma	2018csb1072
Diksha Kapoor	2018csb1084
Ruchika Sharma	2018csb1117
Sweety	2018csb1122