COL-216 Assignment-3

Mohit Thakur 2019CS10373
Sidharth Agarwal 2019CS50661

March 13, 2021

Aim

Develop an interpreter for a subset of MIPS assembly language instructions.

Approach

1. We first declared and array integers for memory, registers, and frequency of instructions(vector in case of frequency). We also declared the integers to store current instruction number and upto which point does the memory has instructions.
2. We also realised that we need to store instructions in bits and thus will need some kind of encoding to store instructions.
3. We decided to split instructions in categories depending on the number of registers and the number of numbers it need to store
4. Then, we read instructions one-by-one and the mapped them to category and further encoded them in integer to store the information.
5. To run the code, we reset the current instruction to 0 and started traversing instruction until it do not crosses end of instruction.

Design Decisions

1. Instead of declaring memory as 1 << 20 bytes we declared it as 1 << 18 integers for ease of storing instructions.
2. Categories of instructions are -
    (a) type_a : add,sub,mul,slt <type> $r1,$r2,$r3
    26-30 bits for unique instruction number assigned
    21-26 bits for register1
    16-21 bits for register2
    11-16 bits for register3

    (b) type_b : bne,beq,addi <type> $r1,$r2,<label/value>
    26-30 bits for unique instruction number assigned
    21-26 bits for register1
    16-21 bits for register2
    0-16 bits used to store integer for instruction number in case of bne and beq and number to be added in addi

(c) type_c : j <type> <label/value>
26-30 bits for unique instruction number assigned
0-26 bits used to store integer for instruction number

(d) type_d : sw,lw <type> $r1,offset($r2)
26-30 bits for unique instruction number assigned
21-26 bits for register1
16-21 bits for register2
0-16 bits for offset

3. We also dealt with case of offset other than 4 in case of lw and sw commands.

4. All instructions formats are very strict, i.e. code throw error when anything unexpected occurs in the file.

Testing

We tested the compiled code on manually written codes.