Computer Architecture - CS2323. Autumn 2024

Lab-3 (RISC-V Assembler)

Name: Ahmik Virani
Roll Number: ES22BTECH11001

Coding Approach

The basic idea was to do 2 iterations of the input. First for identifying any labels and gathering the data by removing the unnecessary commas and spaces and storing this in a data structure. Next iteration I will check the format of instruction and pass that to its corresponding function to return the machine code or error (if any).

Elaborating on the above:

Firstly I initialized all the various registers with their equivalent decimal values and also the functions (funct3 and funct7) and all the formats corresponding to the various instructions.

Then I started by taking the input line by line and storing it in a 2d matrix of strings. Based on the instructions (or the comment), I checked the space and removed any unnecessary characters like a comma or a closing parenthesis and appended it onto the matrix. Also I made sure to track any labels here and update their line numbers as offset from the first line.

In the next iteration, I went along this matrix which I made in the above step and the format. Any error such as unidentified format resulted in error here itself. If there was no error in identifying the format of the instruction, the instruction was sent to the corresponding function to calculate the machine code.

In each function, first the necessary checks were done to ensure that the register values are correct and valid and some wrong input has not been passed. Also errors involving out of range immediate values were also checked here.

If no error was found, I first found the machine code in binary, and then converted it to hexadecimal.

Assumptions made

1. Input test program to be used during evaluation would have a maximum of 50 lines within it.
2. There is only 1 space in between the instruction and the first operand. Also, only 1 space between “,” and second operand and so on. Similarly, one colon after the label and then one space.
3. The program starts on the first character in each line
4. There is only one instruction in each line

5. There are no blank lines in the input file
6. The input file will contain only the real RISC-V instructions, not the pseudo instructions.
7. Immediate/offset values provided with instructions can be assumed to be in decimal only (with a - sign for negative values).
8. A line starting with a semicolon (;) can be treated as a comment and omitted.
9. All the immediate values given will fit into 32 bit integer values.

Test Cases

First I checked:

```
add x1, x2, x3
sub x4, x5, x6
and x7, x8, x9
or x10, x11, x12
xor x13, x14, x15
sll x16, x17, x18
srl x19, x20, x21
sra x22, x23, x24
```

Here, i tested that all the Part 1 instructions are executed correctly and tested x1 - x24 register values are worked on correctly.

Next I tested:

```
addi x25, x26, 2047
andi x27, x28, -2048
ori x29, x30, 0
xori x31, ra, 1
xori sp, gp, -1
slli tp, t0, 0
srli t1, t2, 63
srai x0, zero, 30
```

This ensured some of the alias names are there along with some of the I type instructions.

Next:

```
ld s0, 0(fp)
lw s1, 2047(a0)
lh a1, -2048(a2)
lb a3, 2000(a4)
lwu a5, -261(a6)
lhu a7, -1(s2)
lbu s3, 1(s4)
```

Next:

```
sd s5, -2048(s6)
```

sw s7, 2047(s8)
sh s9, 0(s10)
sb s11, -1(t3)
jalr t4, 1(t5)
sb t6, 1(zero)

This finishes all the alias registers.

Now, since all registers are tested, I just have to test the remaining instructions

L1: xor a5, a3, a7
beq x1, x2, L1
bgeu x1, x2, L1
bltu x1, x2, L1
bge x1, x2, L2
blt x1, x2, L2
bne x1, x2, L2
jal x0, L2
jal x0, L1
L2: add t1, x8, s10

Here I checked B and J type, with both positive and negative offsets.

Finally:

lui x9, 4294963200
lui x9, 0

Here, I understood that using the stoi function is not sufficient, and we must use stoll, which is why is used #define stoi stoll at the top.

<u>Testing Errors:</u>

add x100, x100, x50 -> Invalid register number
addi x1, x1, 2048 -> Out of range immediate
addi x1, x1, -2049 -> Out of range immediate
slli x10, x10, -1 -> Out of range immediate
srli x10, x10, 64 -> Out of range immediate
srai x10, x10, 64 -> Out of range immediate
sfkaf x10, x10, 64 -> Undefined instruction
ld s0, -2049(fp) -> Out of range immediate
ld s0, 2048(fp) -> Out of range immediate
sb s11, -2049(t3) -> Out of range immediate
sb s11, 2048(t3) -> Out of range immediate
jalr x1, -2049(x2)
jalr x1, 2048(x2)
lui x9, 4294963201
lui x9, -1

Below will give an error as label L1 is not defined

```
beq x1, x2, L1
jal x0, L2
L2: add t1, x8, s10
```

Please note that:

1. Not separating operands by a space
2. Not leaving space after a comma
3. Using extra space
4. Adding semicolon in the middle (my code allows semicolon at the start but not middle)
5. Values out of range that the computer cannot store even in long long
6. Not using decimal for immediate/offset values

Will all give errors.