Shiraz University

Assembly

Project Documentation

Part-1

Dr. Esfandiari

TA: Ali Maher

Deadline: *2023 Dec. 17 , Sunday*

Introduction:

The course project has three parts. The first part, is an assembler design project. Assembler is a machine that converts the low-level code of assembly language into machine language or computer 0s and 1s.

For example, the addition command (ADD) is written in machine language 0 and 1, like "0000 0000".

You have to code in the desired language that will do these conversions for us.

"Part 1: Assembler Design"

Example:

```
ADD eax, ecx <======> 01 c8
```

Instructions

Your code needs to support following commands:

ADD, *SUB*, *AND*, *OR*, *INC*, *DEC, XOR, PUSH, POP* and *JMP.*

Consider some tips:

1. Your implementation has to support all registers families.
2. There are several types of "Jump" instruction, but here by "JMP" we mean short jump. And also, your assembler has to support both forward and backward jumps.

This part of project contains three sections.

Section 1 (Register):

In this part of the project, transfers, and operations only take place between registers, which means we have to consider MOD 11. If you read the source I posted below, you will understand what MOD is :]

- You can get help from the site here to understand the conversions of machine commands and how to work.
- You can also use this site to test commands and see their outputs. (You may need a VPN)

Section 2 (Memory):

In the continuation of assembler design, at this stage, we want to add a small part of memory addressing to this machine.

It is enough to add the ability to work with memory to the previous commands. (Only simple register addressing, no displacement, no SIB)

Examples:

```
ADD eax, [ebx] <======> 03 03

          ...

SUB [ecx], edx <======> 29 11
```

But in this part of the project, since the transfers and operations are not only between the registers, we must pay attention to the MOD setting as well.

Section 3 (Jump):

In this section you have to implement the short jump instruction (forward and backward).

Examples:

```
ADD eax, ecx <====> 01 c8
I_am_here: <======> nothing
ADD bx, dx <======> 66 01 d3
JMP I_am_here <===> eb fb
```

Inputs and Outputs:

- Input: The input is a file contains some assembly instructions.
- Output: The output contains from address and byte codes.

Note: Our architecture is little endian.

Sample:

Input:

```
ADD eax, ecx
SUB al, bl
```

Output:

```
0x0000000000000000:  01 C8    (ADD eax, ecx)
0x0000000000000002:  28 D8    (SUB al, bl)
```

Scoring criteria: (300 Pts)

- Correctness (100 pts section 1, 100 pts section 2, 80 pts section 3)
- Report and Documentation
- Comment (First comment, In code comment)
- Clean Code
- Any other creativity will be concerned as bonus.

Note: These criteria may change.

You do not score any if you don't attend for presentation.

• Upload the expected files in a zipped file to the page of the course on Quera.

• Do NOT copy from your classmates' answers :)

• Feel free to ask any questions.

Good Luck!

Ali Maher