Computer Architecture: project 1

environment

```
gcc: 11.3.0
OS: Ubuntu 22.04.1 LTS
Dev on x86_64 Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz
```

Architecture and flow

```
flowchart LR
  input["*.s"] --- tokenizer --> tokens --> syntax_analysis --> semantic_tree --- write ---> binary --> out["*.out"];
```

1. C에서 file I/O를 통해서 `*.s` 파일을 읽어온다.
2. 예약된 instruction, pseudo_instruction에 대한 regex pattern들을 동적으로 생성한다.
3. 파일에서 단어로 해석될 수 있는 단위로 fragment를 만든다.
4. 각 fragment마다 각 token만들고 미리 예약된 regex를 통해서 토큰의 type을 결정한다.
5. 각 token의 타입마다 parse를 통해서 parse된 value들을 token에 넣어준다.
6. 만들어진 token sequence들은 `syntax_analysis`에 들어가고 `bool hierachy(token_t* head, token_t *current)` 에 따른 정책을 통해서 tree가 만들어진다.
7. .data 하위 item들에 대해서 Label의 위치와 data section에 쓰여질 4byte의 리스트를 `write`한다.
8. pseudo_instruction을 instruction으로 바꾸는 작업을 .text 하위 트리에 대해서 진행한다.
9. 계산된 instruction을 통해서 .text label에 대해서 `callback`을 통해서 address를 지정해준다.
10. .text 하위 item들에 대해서 instruction의 작성된 `instruction_set` 을 통해서 backbone을 만든다.
11. 작성된 `field_set` 에 순서, bit offset, bit size, default value 등 따라 하위 노드들을 instruction의 arugment로 바꾸어 리스트를 작성한다. 특정한 `beq`, `bne`, `j` 의 field인 label과, target들은 field type에 맞추어 값이 조정된다.
12. bit mask를 통해서 4btye unsigned int 리스트를 `write` 한다.
13. 작성된 `binary` 리스트를 따라서 `.out` 파일을 작성한다.

Compile

```
make
```

자세한

Run

```
./ascompile <filename>
```

example

```
./ascompile sample.s
```