D:/Coding/codeGen\5\step5.py

```
import os
import re
import requests
import yaml
INPUT_YAML = "mock_out.yaml"
HEADER_FILE = "mock_instruction.h"
OUTPUT_YAML = "generated.yaml"
URL = "https://raw.githubusercontent.com/riscv-software-src/riscv-unified-db/main/spec/std/isa/inst/mock.yaml"
def ensure_yaml_exists():
if not os.path.exists(INPUT_YAML):
print(f"Downloading {INPUT_YAML} from GitHub...")
r = requests.get(URL)
r.raise_for_status()
with open(INPUT_YAML, "w", encoding="utf-8") as f:
f.write(r.text)
def sanitize_macro_name(name):
return re.sub(r'[^A-Z0-9_]', '_', name.upper())
def yaml_to_header_and_yaml(data):
lines_h = []
lines_h.append("/* Auto-generated from YAML */")
lines_h.append("#ifndef MOCK_INSTRUCTION_H")
lines_h.append("#define MOCK_INSTRUCTION_H\n")
lines_h.append(f"#define INST_NAME \"{data['name']}\"")
lines_h.append(f"#define INST_LONG_NAME \"{data['long_name']}\"")
lines_h.append(f"#define INST_DEFINED_BY \"{data['definedBy']}\"")
lines_h.append(f"#define INST_ASSEMBLY \"{data['assembly']}\"")
lines_h.append(f"#define INST_DATA_INDEPENDENT_TIMING {1 if data['data_independent_timing'] else 0}")
lines_h.append(f"#define INST_ENCODING_MATCH \"{data['encoding']['match']}\"")
for var in data['encoding']['variables']:
lines_h.append(f"#define INST_VAR_{sanitize_macro_name(var['name'])}_LOC \"{var['location']}\"")
for k, v in data['access'].items():
lines_h.append(f"#define INST_ACCESS_{k.upper()} \"{v}\"")
lines_h.append("\n#endif /* MOCK_INSTRUCTION_H */")
with open(HEADER_FILE, "w", encoding="utf-8") as f:
f.write("\n".join(lines_h))
with open(OUTPUT_YAML, "w", encoding="utf-8") as f:
yaml.safe_dump(data, f, sort_keys=False)
def main():
ensure_yaml_exists()
with open(INPUT_YAML, "r", encoding="utf-8") as f:
data = yaml.safe_load(f)
```

```
yaml_to_header_and_yaml(data)

if __name__ == "__main__":

main()
```