D:/Coding/codeGen\1 & 2\main.py

```
# 1. Write a Python program that reads at least one of the YAML files in the RISC-V
#    Unified Database project (https://github.com/riscv-software-src/riscv-unified-db)
#    under spec/std/isa/inst.
import requests
import yaml
URL = "https://raw.githubusercontent.com/riscv-software-src/riscv-unified-db/main/spec/std/isa/inst/mock.yaml"
def main():
# Fetch the YAML file
response = requests.get(URL)
response.raise_for_status()
# Parse YAML
mock_data = yaml.safe_load(response.text)
print(yaml.dump(mock_data, sort_keys=False))
if __name__ == "__main__":
main()
# 2. Same Python program then emits the data in the YAML file as a C header file,
#    format of your choosing.
import requests
import yaml
import re
URL = "https://raw.githubusercontent.com/riscv-software-src/riscv-unified-db/main/spec/std/isa/inst/mock.yaml"
def sanitize_macro_name(name):
# Convert to uppercase and replace invalid characters with underscores
return re.sub(r'[^A-Z0-9_]', '_', name.upper())
def yaml_to_c_header(data):
lines = []
lines.append("/* Auto-generated from mock.yaml */")
lines.append("#ifndef MOCK_INSTRUCTION_H")
lines.append("#define MOCK_INSTRUCTION_H\n")
# Basic scalar fields
if "name" in data:
lines.append(f"#define INST_NAME \"{data['name']}\"")
if "long_name" in data:
lines.append(f"#define INST_LONG_NAME \"{data['long_name']}\"")
if "definedBy" in data:
lines.append(f"#define INST_DEFINED_BY \"{data['definedBy']}\"")
if "assembly" in data:
lines.append(f"#define INST_ASSEMBLY \"{data['assembly']}\"")
if "data_independent_timing" in data:
val = 1 if data["data_independent_timing"] else 0
lines.append(f"#define INST_DATA_INDEPENDENT_TIMING {val}")
```

```
# Encoding
if "encoding" in data:
enc = data["encoding"]
if "match" in enc:
lines.append(f"#define INST_ENCODING_MATCH \"{enc['match']}\"")
if "variables" in enc:
for var in enc["variables"]:
macro_name = sanitize_macro_name(f"INST_VAR_{var['name']}_LOC")
lines.append(f"#define {macro_name} \"{var['location']}\"")
# Access
if "access" in data:
for k, v in data["access"].items():
macro_name = sanitize_macro_name(f"INST_ACCESS_{k}")
lines.append(f"#define {macro_name} \"{v}\"")
lines.append("\n#endif /* MOCK_INSTRUCTION_H */")
return "\n".join(lines)
def main():
response = requests.get(URL)
response.raise_for_status()
mock_data = yaml.safe_load(response.text)
header_content = yaml_to_c_header(mock_data)
with open("mock_instruction.h", "w") as f:
f.write(header_content)
print("C header file generated: mock_instruction.h")
if __name__ == "__main__":
main()
```