SS LAB 4

SAHIL BONDRE: U18CO021

Generate variant-I and variant-II representation for multiplication of two numbers.

requirements.txt

```
Package    Version
---------- -------
pip        21.0.1
setuptools 53.0.0
tabulate   0.8.7
termcolor  1.1.0
wheel      0.36.2
```

index.py

```
import sys
import re
from termcolor import colored
from tabulate import tabulate

if len(sys.argv) != 2:
    print("Usage: python index.py <file-name>")
    exit(1)

file_name = sys.argv[1]

symbol_table = []  # no, symbol, address
literal_table = []  # no, literal, address
pool_table = []  # no
ir = []  # line, var1, var2
variant1 = []  # str
variant2 = []  # str

mot = {
    "MOVER": "01",
    "MOVEM": "02",
    "ADD": "03",
    "SUB": "04",
```

```
    "MULT": "05",
    "DIV": "06",
    "BC": "07",
    "COMP": "08",
    "PRINT": "09",
    "READ": "10"
}

pot = {
    "START": "01",
    "END": "02"
}

dl = {
    "DS": "01",
    "DC": "02"
}

registers = {
    "A": "01",
    "B": "02",
    "C": "03",
    "D": "04",
}


def update_label_to_symbol_table(label):
    for row in symbol_table:
        if row[1] == label:
            return row[0]
    symbol_table.append([len(symbol_table) + 1, label, -1])
    return len(symbol_table)


def update_literal_to_literal_table(literal):
    for row in literal_table:
        if row[1] == literal:
            return row[0]
    literal_table.append([len(literal_table) + 1, literal, -1])
    return len(literal_table)


def add_token_to_table(token):
    if "REG" in token:
        register = token[0]
        variant1[-1] += f" {registers[register]}"
```

```
        variant2[-1] += f" {token}"
        return
    if token.startswith("="):
        variant1[-1] += f" (L,
{update_literal_to_literal_table(token)})"
        variant2[-1] += f" (L,
{update_literal_to_literal_table(token)})"
        return

    variant1[-1] += f" (S, {update_label_to_symbol_table(token)})"
    variant2[-1] += f" {token}"


address = 100
with open(file_name) as f:
    for num, line in enumerate(f, 1):
        tokens = line.split()
        is_label_def = not bool(re.match(r'\s', line))
        is_literal_def = tokens[0].startswith("=")
        op = tokens[0]

        ir.append([line.split("\n")[0]])
        variant1.append(f"{address}> ")
        variant2.append(f"{address}> ")

        if num == 1:
            address = int(tokens[1]) - 1
            variant1[-1] += f"(AD, {pot[op]}) - (C, {tokens[1]})"
            variant2[-1] += f"(AD, {pot[op]}) - (C, {tokens[1]})"
        elif is_label_def:
            label = tokens[0]
            for row in symbol_table:
                if row[1] == label:
                    row[2] = address
                    variant1[-1] += f"(DL, {dl[tokens[1]]}) - (C,
{tokens[2]})"
                    variant2[-1] += f"(DL, {dl[tokens[1]]}) - (C,
{tokens[2]})"
                    break
        elif is_literal_def:
            literal = tokens[0]
            for row in literal_table:
                if row[1] == literal:
                    row[2] = address
                    lit = tokens[0]
                    num = [int(s) for s in lit.split("\"") if
```

```
s.isdigit()][0]
                    variant1[-1] += f"(AD, 05) - (C, {num})"
                    variant2[-1] += f"(AD, 05) - (C, {num})"
                    break
        elif op == "ORIGIN":
            address = int(tokens[1]) - 1
        elif op == "LTORG":
            address -= 1
        elif op == "END":
            variant1[-1] += "(AD, 02)"
            variant2[-1] += "(AD, 02)"
        else:
            variant1[-1] += f"(IS, {mot[op]})"
            variant2[-1] += f"(IS, {mot[op]})"
            if len(tokens) >= 2:
                operand = tokens[1]
                add_token_to_table(operand)
            if len(tokens) >= 3:
                operand = tokens[2]
                add_token_to_table(operand)

        address += 1


def generate_pool_table(literal_table):
    pool_table = []
    for i, r in enumerate(literal_table):
        if i == 0:
            pool_table.append([r[0]])
        elif r[2] != 1 + literal_table[i - 1][2]:
            pool_table.append([r[0]])
    return pool_table


pool_table = generate_pool_table(literal_table)

print(colored("Symbol Table", attrs=["bold"], color="blue"))
print(tabulate(symbol_table, headers=[
      colored("No.", color="yellow"), colored("Symbol", color="yellow"),
colored("Address", color="yellow")], tablefmt="fancy_grid"))
print()

print(colored("Literal Table", attrs=["bold"], color="blue"))
print(tabulate(literal_table, headers=[
      colored("No.", color="yellow"), colored("Literal",
color="yellow"), colored("Address", color="yellow")],
```

```
tablefmt="fancy_grid"))
print()

print(colored("Pool Table", attrs=["bold"], color="blue"))
print(tabulate(pool_table, headers=[
      colored("No.", color="yellow")], tablefmt="fancy_grid"))
print()

print(colored("Intermidiate Representation", attrs=["bold"],
color="blue"))

for i in range(len(variant1)):
    ir[i].append(variant1[i])
    ir[i].append(variant2[i])
print(tabulate(ir, headers=[
      colored("Code", color="yellow"), colored("Variant 1",
color="yellow"), colored("Variant 2", color="yellow")],
tablefmt="fancy_grid"))
```

file.asm

```
  START 200
  READ X
  MOVER AREG, X
  MOVER BREG, ="5"
  MULT AREG, BREG
  MOVEM AREG, Y
  PRINT Y
  LTORG
    ="5"
  ORIGIN 500
X DS 7
Y DS 12
  END
```

Symbol Table

No.	Symbol	Address
1	X	500
2	Y	501

Literal Table

No.	Literal	Address
1	="5"	206

Pool Table

No.
1

Intermidiate Representation

Code	Variant 1	Variant 2
START 200	100> (AD, 01) - (C, 200)	100> (AD, 01) - (C, 200)
READ X	200> (IS, 10) (S, 1)	200> (IS, 10) X
MOVER AREG, X	201> (IS, 01) 01 (S, 1)	201> (IS, 01) AREG, X
MOVER BREG, ="5"	202> (IS, 01) 02 (L, 1)	202> (IS, 01) BREG, (L, 1)
MULT AREG, BREG	203> (IS, 05) 01 02	203> (IS, 05) AREG, BREG
MOVEM AREG, Y	204> (IS, 02) 01 (S, 2)	204> (IS, 02) AREG, Y
PRINT Y	205> (IS, 09) (S, 2)	205> (IS, 09) Y
LTORG	206>	206>
="5"	206> (AD, 05) - (C, 5)	206> (AD, 05) - (C, 5)
ORIGIN 500	207>	207>
X DS 7	500> (DL, 01) - (C, 7)	500> (DL, 01) - (C, 7)
Y DS 12	501> (DL, 01) - (C, 12)	501> (DL, 01) - (C, 12)
END	502> (AD, 02)	502> (AD, 02)