# Code Generation

SHRI VISHNU ENGINEERING COLLEGE FOR WOMEN

BHIMAVARAM

April 9, 2022

# BATCH - 5



1. 20B01A05H8-V.RAKSHITHA-CSE
2. 20B01A05H9-V.RESHMA-CSE
3. 20B01A1215-B.BINDU SRI-IT
4. 20B01A1216-B.HARITHA-IT
5. 20B01A04C0-P.KEERTHI-ECE

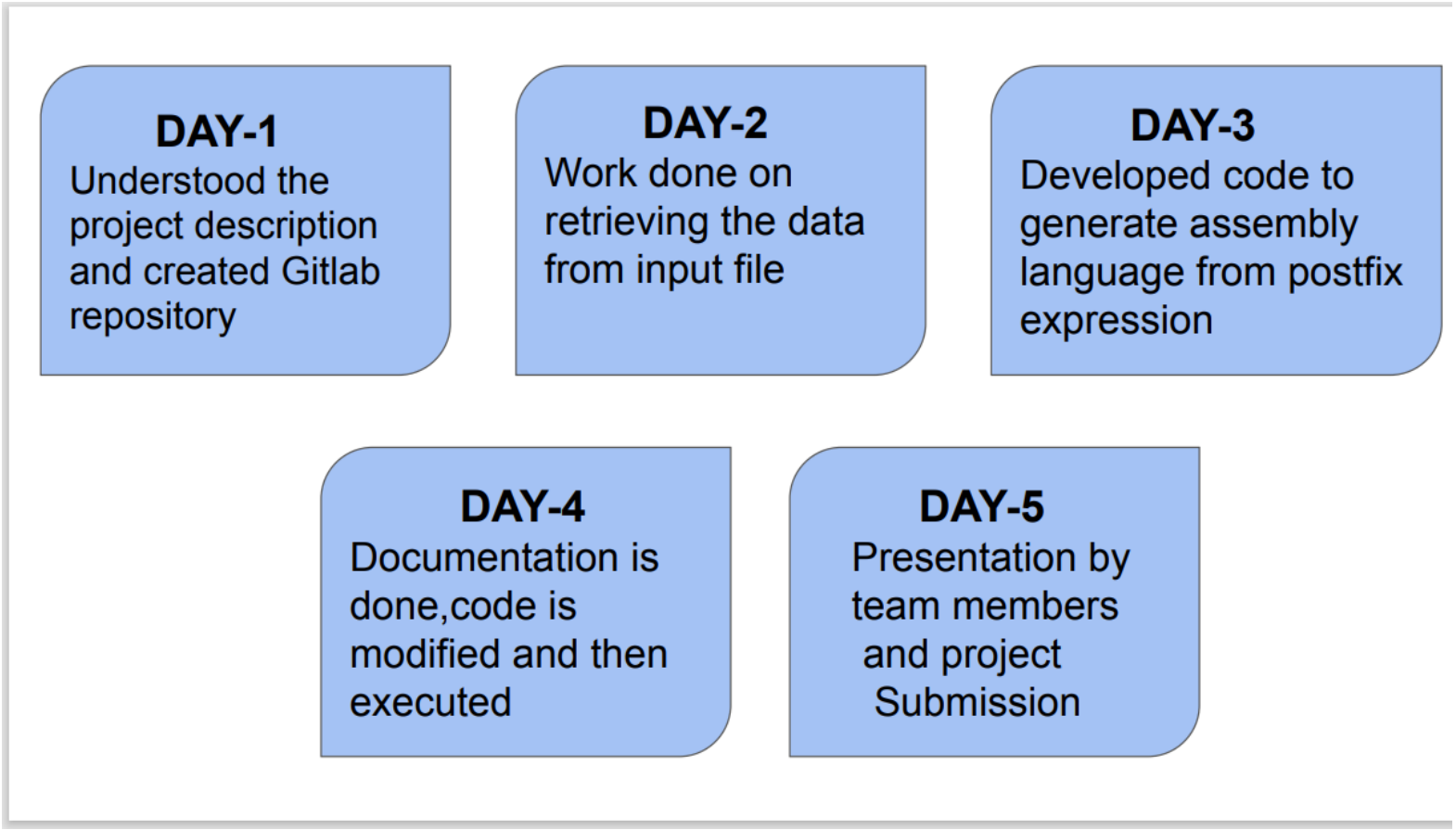
# INTRODUCTION

Code Generation is building a translator to convert the postfix arithmetic expression into assembly language code

# DESCRIPTION

Code Generator is a backend translator for an SIC machine. Input to the translator will be arithmetic expressions in postfix form and the output will be assembly language code.

# FLOW DIAGRAM



# MEMBERS CONTRIBUTION

- Bindu Sri - Installed related software and worked on retrieving the input from input.txt file which consists of postfix expression.
- Keerthi - worked on modifying the inputdata to pass it to the function which generates assembly language.
- Reshma - Worked on generating assembly language code as output.
- Rakshitha - Worked on generating assembly language code as output.
- Haritha - Worked on appending the obtained output in the output.txt file.

# CHALLENGES



- for retrieving the input data to the main file
- understanding and implementing the assembly language

# HOW CHALLENGES OVERCAME

- created a seperate input.text file to retrieve it into the main file
- gaining the knowledge about assembly language through internet

# LEARNINGS



- Team collaboration.
- Preparing presentations in LaTeX.
- Building projects with Python language.
- Push files into repository(Gitlab).

# TECH STACKS

1. Python Programming
2. Pycharm
3. GitLab
4. GitBash

# CODE STATS



- Conditional statements and loops (while, for ,if-elif).
- Functions-write(),isdigit(),close(),open(),strip(),split()

# CODE STATS

- Our code consists of 55 lines.

```
def printLineCode(i):
    var = open('output.txt', 'a')
    temps = 1
    line = i
    x = 0
    if len(line) == 1:
        if line[0].isdigit():
            var.write('L #' + line[0] + '\n\n')
        else:
            var.write('L ' + line[0] + '\n\n')
    else:
        while len(line) != 1:
            if line[x+2] in "+-*/":
                op = line[x+2]
                num1 = '' + line[x]
                num2 = '' + line[x+1]
                placehold = ''

                if num2.isdigit():
                    placehold = '#'

                if num1.isdigit():
                    var.write('L #' + num1 + '\n\n')
                else:
                    var.write('L ' + num1 + '\n\n')

                if op == '+':
                    var.write('A ' + placehold + num2 + '\n\n')
                elif op == '-':
                    var.write('S ' + placehold + num2 + '\n\n')
                elif op == '*':
                    var.write('M ' + placehold + num2 + '\n\n')
                elif op == '/':
                    var.write('D ' + placehold + num2 + '\n\n')
                else:
                    var.write('N ' + placehold + num2 + '\n\n')

                var.write('ST $' + str(temps) + '\n\n')
                line[x] = '$' + str(temps)
                temps += 1
                line.pop(x + 1)
                line.pop(x + 1)
                x = 0
            else:
                x += 1
        var.close()

open('output.txt', 'w').close()
file = open('input.txt', 'r')
for line in file:
    line = line.strip()
    line = line[0:len(line)]
    tokens = line.split(" ")
    printLineCode(tokens[0:len(line)])
file.close()
```

# output

output.txt

```
1   L A
2
3   A B
4
5   ST $1
6
7   L C
8
9   A D
10
11  ST $2
12
13  L E
14
15  A F
```

# REPO

- GitLab link

https://gitlab.com/vrakshitha2468/codegeneration-wise-batch-5.git


Thank you
Play