CS3300: Compiler Design **Odd Sem**

Lab #10

Code Generation Deadline: 07/10/2018, 11:55PM

Important Instructions

- Program should read from stdin and write to stdout.
- Follow submission guidelines carefully.
- Violating any of the above instructions will incur penalty.

Task

Write a C/C++ program using lex/yacc to implement function overloading in C. The program needs to generate assembly code for the given input program. A grammar file has been included, which contains all the constructs that has to be implemented.

Input

A valid program i.e. a program which can be generated from the given context free grammar.

Output

Print the assembly code for the given input program.

Notes

- There won't be any overloading for main function.
- An expression will contain either int or float, not both.
- Global and local variables will have different names.
- No two variables in a function scope will have same name.

Evaluation process

- Run your program to generate assembly code.
- Run gnu assembler to generate executable.
- Run executable and compare the outputs with the actual outputs.

Examples

Example 1 - Overloading based on the number of arguments

Input Program

```
int add(int x, int y) {
    return x+y;
}

int add(int x, int y, int z) {
    int sum;
    sum = (x + y) + z;
    return sum;
}

int main() {
    int sum1;
    int sum2;
    int a;
    int b;
    a = 2;
    b = 3;
    sum1 = add(a, b);
    sum2 = add(a, b, 4);
    printf("%d\n", sum1);
    printf("%d\n", sum2);
    return 0;
}
```

Example 2 - Overloading based on the order of types of the arguments

Input Program

```
float func(int x, float y) {
    if (x < 0) {
        y = y * 0.5;
    }
    return y;
}

float func(float x, int y) {
    if(y < 0) {
        x = x * 2.0;
    }
    return x;
}


int main() {
    float val1;
    float val2;
    val1 = func(2, 3.2);
    val2 = func(2.1, 3);
    printf("%f\n", val1);
```

```
    printf("%f\n", val2);
    return 0;
}
```

Submission

Submit a tar.gz file with filename as <ROLLNO >.tar.gz (eg. CS12B043.tar.gz) containing the following structure:

- CS12B043 <directory>
  - *.l
  - *.y
  - Makefile

The Makefile should generate an executable **a.out**.