

C to Assembler

Aarón Arias Pérez

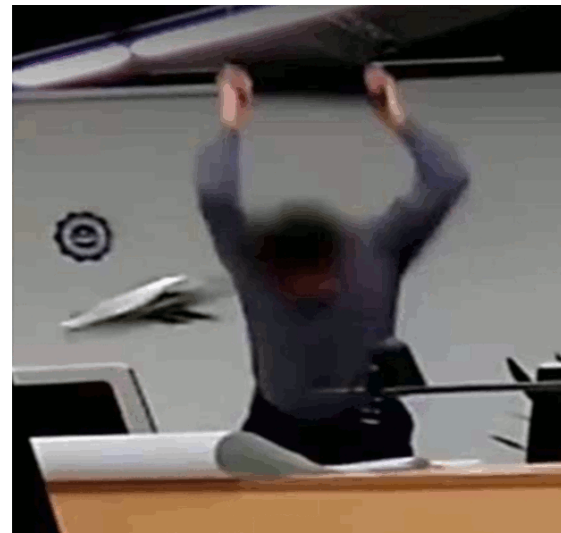
José Joaquín Arias Gómez-Calcerrada

What do we need to do?

- Ascendent grammar
- Generate an intermediate AST
- Generate assembler code from that AST

Problems

- Ambiguity
- Labels
- Offset
- Tables



```
int fact(int n) {  
    if(n<=1) {  
        n=1;  
    }  
    else {  
        return n*fact(n);  
    }  
    return fact(n);  
}  
  
int numero;  
  
int main() {  
    scanf("%d", &numero);  
    printf("El factorial = %d\n", fact(numero));  
    return 0;  
}
```

Tables
Global variables
Local variables
Argument variables

Variable name
variables
variables_locales
variables_args

```
1. %{
2.
3. #include <iostream>
4. #include <string>
5. #include <vector>
6. #include <map>
7. #include <fstream>
8. #include <stack>
9. #include "TablaVariables.h"
10. #include "node.h"
11.
12. using namespace std;
13.
14. int yyerror(const char* msj);
15. int yylex(void);
16.
17. TablaVariables variables;
18. TablaVariables variables_locales;
19. TablaVariables variables_args;
20.
21. vector<string> call_pushs;
22.
23. stack<int> etiquetas_while;
24. stack<int> etiquetas_if;
25. stack<int> etiquetas_and;
26. stack<int> etiquetas_or;
27. stack<int> etiquetas_not;
28. stack<int> etiquetas_eq;
29. stack<int> etiquetas_neq;
30. stack<int> etiquetas_gte;
31. stack<int> etiquetas_gt;
32. stack<int> etiquetas_lte;
```

Return node

- If \$2 is a variable, we check the tables
- Checking order:
 - Local variables
 - Argument variables
 - Global variables

Denoting the offset

Labels

- Usage of a stack for each situation
 - "While" stack
 - "If" stack
 - "And" stack
 - Etc..
- Zero nesting problems

Demo time

