Generate x86 assembly code for programs written using the following grammar. The assembly code should be output to a file, which we will assemble using the *gcc* assembler and execute on the machine. Use the following commands:

```
grun MyGram prog < <path to test case> > <assembly file e.g. test.s>
gcc -s <path to assembly file>
./a.out
class Program { <field_decl>* <method_decl>* }
<field decl> -> <type> (<id> | <id> [ <int literal> ] ) ( , <id> | <id> [ <int literal> ] )*;
<field_decl> -> <type> <id> = teral> ;
<method_decl> -> ( <type> | void ) <id> ( ((<type> <id>) ( , <type> <id>)*)? ) <block>
<blook> -> { <var_decl>* <statement>* }
<var decl> -> <type> <id> ( , <id>)*;
<type> -> int | boolean
<statement> -> <location> <assign op> <expr> ;
<statement> -> <method call>;
<statement> -> if ( <expr> ) <block> ( else <block> )?
<statement> -> for <id> = <expr> , <expr> <block>
<statement> -> return ( <expr> )?;
<statement> -> break;
<statement> -> continue;
<statement> -> <block>
<assign op> -> =
<assign_op> -> +=
<assign_op> -> -=
<method call> -> <method name> ( (<expr> ( , <expr> )*)? )
<method_call> -> callout ( <string_literal> ( , <callout_arg> )* )
<method name> -> <id>
<location> -> <id>
<location> -> <id> [ <expr> ]
<expr> -> <location>
<expr> -> <method call>
<expr> -> teral>
<expr> -> <expr> <bin op> <expr>
<expr> -> - <expr>
<expr> -> ! <expr>
<expr> -> ( <expr> )
<callout arg> -> <expr> | <string literal>
<bin_op> -> <arith_op> | <rel_op> | <eq_op> | <cond_op>
<arith_op> -> + | - | * | / | %
<rel op> -> < | > | <= | >=
<eq op> -> == | !=
<cond_op> -> && | ||
-> <int literal> | <char literal> | <bool literal>
```

```
<id> -> <alpha> <alpha_num>*
<alpha> -> [a-zA-Z_]
<alpha_num> -> <alpha> | <digit>
<digit> -> [0-9]
<hex_digit> -> <digit> | [a-fA-F]
<int_literal> -> <decimal_literal> | <hex_literal>
<decimal_literal> -> <digit> <digit>*
<hex_literal> -> 0x <hex_digit> <hex_digit>*
<bool_literal> -> true | false
<char_literal> -> "<char>"
<string_literal> -> "<char>""
```