# Mini-Relatório - Projeto 2
# Linguagens formais e autómatos
# Grupo 3

**David Moreira de Almeida, Diogo Daniel Soares Ferreira,**
**Francisco Miguel Matos da Cunha, José Pedro Baião Castanheira**

1

## 1. Indicação do Trabalho Realizado

O trabalho realizado por cada um foi (de 0 a 400):

- David Moreira - 83,3
- Diogo Ferreira - 150
- Francisco Cunha - 83,3
- José Castanheira - 83,3

## 2. Distribuição pelos elementos do grupo

O trabalho acabou por sofrer alterações em todas as partes por todos os elementos do grupo, devido à vontade de todos de contribuir para o projeto e à solidariedade.

No entanto, as divisões iniciais de trabalho foram:

- *Parser* - Diogo Ferreira e José Castanheira
- *Lexer* - Diogo Ferreira e José Castanheira
- Alterações no *main* - David Almeida
- Ficheiros de teste - Francisco Cunha e David Almeida
- Funções *generateLsm* - José Castanheira, Francisco Cunha, David Almeida e Diogo Ferreira

## 3. Pequena descrição do trabalho realizado

Tal como pedido, não iremos fornecer uma descrição exaustiva do trabalho realizado, apenas alguns pormenores que poderão diferir dos outros grupos e que achamos importante apontar.

Foram criadas mais classes que serão nós da árvore sintática devido à necessidade de acrescentar particularidades à linguagem gerada.

Sempre que é necessário em argumentos de funções, operações aritméticas e outras operações, é feito o *cast* para o tipo necessário. Sempre que existe *downcast*, o compilador gera um warning para o utilizador. Também é possível efetuar *cast* explícito, escrevendo o tipo dentro de parêntesis, como efetuado em baixo.

```
1 procedure main(){
2   float b, c=2.35;
3
4   b=(int) c;
5 }
```

Foram também acrescentadas instruções *bitwise* & (*and*), $\|(or), \sim$ $(not), \ll$ (*shift left*), $\gg$ (*shift right*)$e \bigwedge(xor)$.

Para simplificar a escrita de programas, decidimos acrescentar instruções unárias típicas na linguagem *c*. Portanto, adicionámos também as instruções ++, –, +=, -=, /=, %= e *=. Todas necessitam de ter uma variável numérica à sua esquerda para serem válidas.

A inicialização dos *arrays* pode ser feita de várias formas diferentes. Todas as descritas abaixo são válidas.

```
1 procedure main(){
2   int a[4], [4]b, c={1, 3, 4};
3 }
```

Optámos por não adicionar *arrays* não-inicializados à linguagem, nem *arrays* inicializados com tamanho e conteúdo simultaneamente, para simplicidade do projeto.

É importante referir que as funções *printInt* e *readInt* estão escritas em linguagem *asm* nas funções *generatelsm* de cada una delas, para poderem funcionar corretamente. A função *printStr* é realizada com base nas instruções já adicionadas ao projeto 1 (*asm*).

Foi acrescentada uma *symbol table* hierárquica parar ser usada dentro de blocos como *for* ou *if*. Assim, é possível declarar variáveis dentro de blocos interiores que deixam de ser válidas para o *scope* global, sem perder as variáveis já anteriormente declaradas.

Decidimos também adicionar à linguagem *procedures* sem argumentos de entrada, sem argumentos de retorno e com a mesma *symbol table* usada anteriormente antes da chamada à função.

Por opção do grupo, se colocarmos a instrução *break* fora de um *loop*, sintaticamente não gera nenhum erro. No código gerado em *asm*, o salto é ignorado.

Para testar todo o projeto, tentámos construir o maior número de ficheiros de exemplo possível, alguns com código específico básico, outros com código mais complexo. Temos 40 ficheiros de exemplo para teste na pasta *examples*. Todos são corretamente gerados e todos correm sem problemas na máquina virtual do projeto 1.

## 4. Instruções para compilar ficheiros em linguagem s16 para ASM

Para gerar código, enviar como arguento -g". Assim, gerará código para a linha de comandos.

Se quiser gerar código diretamente para um ficheiro, é necessário enviar como argumento -o". Assim, criará um ficheiro com o mesmo nome que o ficheiro de entrada, mas com extensão ".s".

Se quiser especificar o nome do ficheiro como argumento, envie o nome do ficheiro junto à instrução -o".

Exemplo: Ficheiro de output desejado: test.s

Comando: ./s16 inputfile -g -otest.s