Geração de Código
Parte I
Roteiro para atividades

Este roteiro abre os trabalhos para a fase de geração de código do compilador.
Para implementar a geração de código, é importante que se conheça a arquitetura alvo, que nesse caso é a MicroJava VM (máquina virtual MicroJava). O roteiro mostra diversos exemplos de como comandos MJava podem ser traduzidos para instruções assembly MJava.

Instruções:

1. Estude a especificação da máquina virtual MicroJava, no Apêndice B do documento *mJava.pdf*.
2. Obtenha o projeto *mJVM.zip*, que contém:
    a. um montador (*Assembler.java*);
    b. um interpretador para instruções da máquina virtual MicroJava (*MJ.java*);
    c. uma classe para disparar montagem seguida de execução (*AssembleAndRun.java*);
    d. uma classe para disparar montagem seguida de depuração (*AssembleAndDebug.java*);
    e. alguns exemplos.
3. Estude os arquivos *exemplo1*, *exemplo2* e *exemplo3*, com extensão *mjava* e *mjasm*. Os arquivos com extensão *mjava* contêm código escrito na linguagem MicroJava. Quando o seu compilador estiver completo, deverá ser capaz de compilar esses programas. Cada arquivo com extensão *mjasm* corresponde a uma compilação do programa de mesmo nome, com extensão *mjava*. Compare os arquivos *mjava* com os seus respectivos *mjasm*, e observe como as construções podem ser traduzidas.
4. Execute o programa *AssembleAndRun* ou *AssembleAndDebug*, dando como entrada os códigos *assembly* dos exemplos fornecidos (quando o programa pedir para digitar o arquivo, basta fornecer nome sem extensão – por exemplo: “exemplo1”). Esses programas disparam o montador e depois executam o código objeto gerado, sendo que *AssembleAndDebug* ainda exibe informações para depuração, incluindo a instrução sendo executada e o estado da pilha de expressões. Observe os resultados exibidos.
5. Construa um programa *assembly* que corresponda a uma possível compilação do programa MicroJava apresentado a seguir. Usando as ferramentas fornecidas, traduza o programa *assembly* para código de máquina e o execute (*AssembleAndRun* ou *AssembleAndDebug*), para testar se foi construído corretamente.
6. Envie o programa *assembly* desenvolvido, usando a plataforma Google Classroom.

```
program P

class C {
  int v;
  C prox;
}

// não tem variáveis globais


{


// função f tem 2 parâmetros e 4 variáveis, no total
C f(int a, int b)
C x1, x2;
{
  x1 = new C; // um objeto da classe C gasta
              // 8 bytes para ser armazenado
  x1.v = a;
  x1.prox = new C; // um objeto da classe C gasta
                   // 8 bytes para ser armazenado
  x2 = x1.prox;
  x2.v = b;
  x2.prox = null; // null pode ser tratado como
                  // constante inteira 0 (zero)
  return x1;
}


void main()
C x;
{
  x = f(1,2);
  while (x != null) {  // null pode ser tratado como
                       // constante inteira 0 (zero)
    print(x.v, 4);
    x = x.prox;
  }
}

}
```