Compiladores – 2016/2
Roteiro de Laboratório 12

1 Objetivos

O objetivo deste laboratório é implementar um compilador da linguagem TINY que gera código *assembly* para a Tiny Machine (TM).

2 TM – Tiny Machine

A TM é uma arquitetura extremamente simples que pode ser usada como código alvo para exercitarmos a geração de código de um compilador. A TM não existe fisicamente, portanto ela é simulada em *software*.

A TM possui 1024 posições de memória de instruções (`iMem`) e 1024 posições de memória de dados (`dMem`), além de 64 registradores. Esses valores pode ser modificados através de macros. Veja o arquivo `tm.c`. Todos os registradores são de uso geral, exceto o registrador 0, que é o *PC (program counter)*.

A TM possui o seguinte conjunto de instruções:

Instrução	**Efeito**
`HALT`	Termina a execução
`IN r`	`r <- stdin`
`OUT r`	`r -> stdout`
`ADD r, s, t`	`r <- s + t`
`SUB r, s, t`	`r <- s - t`
`MUL r, s, t`	`r <- s * t`
`DIV r, s, t`	`r <- s / t`
`LD r, o, s`	`r <- dMem[s + o]`
`LDA r, o, s`	`r <- s + o`
`LDC r, c`	`r <- c`
`ST r, o, s`	`dMem[s + o] <- r`
`JON r, o`	`PC <- PC + o, if r < 0`
`JNZ r, o`	`PC <- PC + o, if r != 0`
`JMP r, a`	`PC <- a`

aonde `r, s, t` são registradores, sempre no formato `r[1-63]`; `o` é um *offset*; `c` é uma constante e `a` é um endereço absoluto na memória de instruções.

Detalhes da implementação do simulador TM:

- Comentários são indicados por `;`. Todos comentários são de linha, não existem comentários de bloco.
- Linhas vazias ou que comecem com um comentário são ignoradas.
- Os *opcodes* são todos em maiúsculas e devem aparecer no início da linha.
- Os operandos devem aparecer na coluna 5 em diante.

Para mais detalhes, veja o código no arquivo `tm.c`. Para compilar o simulador, digite `make tm`. Um exemplo de um código para TM:

```
; This program reads an integer and output its factorial.

IN   r1          ; r1 <- read from stdin
JON  r1, 6       ; Jump 6 instructions ahead if r1 < 0
LDC  r2, 1       ; r2 <- 1
LDC  r3, 1       ; r3 <- 1
MUL  r2, r2, r1  ; r2 <- r2 * r1
SUB  r1, r1, r3  ; r1 <- r1 - r3
JNZ  r1, -3      ; Jump 3 instructions back if r1 != 0
OUT  r2          ; r2 -> stdout
HALT
```

3 Geração de Código

Implemente um gerador de caminha recursivamente na AST e emite o código para cada nó. Baseie-se no comportamento do interpretador do laboratório anterior para inferir o código que deve ser gerado em cada caso.

Os arquivos de *template* disponibilizados no AVA realizam a construção da AST. Também está disponível o código do simulador TM. Faça o *download* do código e entenda o seu funcionamento. A tarefa deste laboratório é desenvolver o código do arquivo `code.c`.

4 Passos para realização do laboratório

1. Faça o *download* dos arquivos de *template* no AVA.
2. Implemente a função `emit_code()`.
3. Teste o seu compilador executando o código gerado no simulador da TM. Os programas de entrada são os mesmos dos laboratórios anteriores.

Observações importantes:

- A linguagem TINY tratada aqui é a especificação original, isto é, sem as extensões descritas no Laboratório 09.
- Uma implementação de referência para esse laboratório será disponibilizado pelo professor em um futuro próximo. No entanto, você é *fortemente* encorajado a realizar a sua implementação completa antes de ver uma solução em outro lugar.