Trabalho de Compiladores - Parte 1

Victor Hugo de Oliveira Gomes - 21852452
Nilson Andrade dos Santos Júnior - 21853906
Aldemir Silva - 21951671

Para fazer as análises sintáticas e léxicas, além da geração do código intermediário e Assembly MIPS

```
python3 main.py <nome_do_arquivo>
```

Passe como argumento o nome do arquivo contendo o código escrito em MiniC.

Assim que o *script* executar, dois arquivos serão gerados: `nome_do_arquivo.i` e `nome_do_arquivo.s`, que são, respectivamente, o arquivo que contém a conversão do código escrito em MiniC para a representação de 3 endereços e o arquivo que contém a conversão do código de 3 endereços em Assembly MIPS.

Traduzindo Código Intermediário para Assembly MIPS

Instrução de atribuição

1. **Atribução com constantes**

```
a = 1
```

```
addi    r0, r1, #1
```

2. **Atribuição simples**

```
a = b
```

```
add     r0, r1, r2
```

Operações aritiméticas

1. **Adição**

```
a = b + c
```

```
add     r0, r1, r2
```

2. **Subtração**

```
a = b - c
```

```
sub     r0, r1, r2
```

3. **Multiplicação**

```
a = b * c
```

```
mul     r1, r2
mflo    r0
```

4. **Divisão**

```
a = b / c
```

```
div     r1, r2
mflo    r0
```

5. **Módulo**

```
a = b % c
```

```
div     r1, r2
mfhi    r0
```

Operações Lógicas

1. **Menor que**

```
a = b < c
```

```
slt     r0, r1, r2
```

2. **Maior que**

```
a = b > c
```

```
slt     r0, r2, r1
```

3. **Menor ou igual**

```
a = b <= c
```

```
slt     r0, r2, r1
xori    r0, r0, #1
```

4. **Maior ou igual**

```
a = b >= c
```

```
addi    r3, r2, #1
slt     r0, r1, r3 # (b >= c) é equivalente a (b < c + 1)
```

5. **And**

```
a = b && c
```

```
# Nesse ponto, ambos os registradores sendo testados vão ser 0 ou 1,
um 'and' bitwise entre eles retorna 0 se eles forem iguals (ambos 1 ou
```

```
ambos 0) ou 1, se forem diferentes.
and     r0, r1, r2
```

6. **Or**

```
a = b || c
```

```
# Nesse ponto, ambos os registradores sendo testados vão ser 0 ou 1,
um 'or' bitwise entre eles retorna 0 se pelo menos um deles for 1 e 0
se os dois forem 0.
or      r0, r1, r2
```

7. **Igual a**

```
a = b == c
```

```
xor     r0, r1, r2
```

8. **Diferente de**

```
a = b != c
```

```
xor     r0, r1, r2
```

Controle de fluxo

1. **Condicional**

```
if t goto label_1, goto label_2
```

```
bnez    r0, label_1
b       label_2
```

2. **Goto**

```
goto label_1
```

```
j   label_1
```

3. **Label**

```
label_1:
```

```
label_1:
```

Examplo

Código em MiniC

```
int main() {
    int a;
    int b;

    a = 10;
    b = 15;

    while (a <= b) {
        ++a;
    }
    return b;
}
```

Código intermediário

```
main():
a = 10
b = 15
L0:
t0 = a <= b
if t0 goto L1, goto L2
L1:
a = a + 1
goto L0
L2:
t1 = b
return t1
```

Assembly MIPS

```
    .code
    addi    r2, r0, #10
    addi    r3, r0, #15
L0:
    slt     r4, r3, r2
    xori    r4, r4, #1
    bnez    r4, L1
    b       L2
L1:
    addi    r2, r2, #1
    j       L0
L2:
    add     r5, r0, r3
```