

Trabalho Prático Compiladores

Etapa 6 - Geração de Assembly

Ricardo Rodrigues Ehlert - 00313284
Vitor Camargo de Moura - 00315212

Introdução

Na etapa 6 do trabalho prático, a proposta era a geração de código assembly a partir das TACs geradas na etapa 5, para isso foi feito o acompanhamento das aulas presenciais, bem como a visualização dos vídeos disponíveis na página da disciplina.

Os comandos empregados foram multiplicação (`*`), divisão (`/`), soma (`+`), subtração (`-`), menor que (`<`), maior que (`>`), assinalamento (`<-`) e `print`. Além disso foram implementados as TACs de `JFALSE`, `LABEL` e `JMP`, possibilitando assim a implementação de funções de controle de fluxo.

Toda a implementação foi feita a partir da geração de código Assembler em um programa normal na linguagem C e compilado com o comando `"gcc -S test1.c"` gerando um arquivo `"test1.s"` que, ao ser analisado nos possibilitou entender quais comandos eram utilizados para cada função, os comandos então foram separados em um arquivo `"testp.s"` para depois serem transferidos para a geração de ASM

Testes

1.

```
1 |
2 | int a(2);
3 | int b(777);
4 | int c(111);
5 |
6 |
7 | int main (){
8 |
9 |     >> print(a);
10 | >> b <- 8;
11 | >> print(b);
12 | >> while(a<b){
13 | >>     >> a <- a+1;
14 | >>     >> print(a);
15 | >> };
16 |
17 | }
18 |
19 |
```

Código 1

No código 1 temos a implementação de um controle de fluxo com `while`, a ideia é que `a`, iniciado em 2, se incremente até que chegue em `b`, assinalado em 8 na linha 10.

```
TAC(TAC_BEGINFUNC,main,0,0);
TAC(TAC_PRINT,0,a,0);
TAC(TAC_ASSIGN,b,8,0);
TAC(TAC_PRINT,0,b,0);
TAC(TAC_LABEL,sPeRWeIRdFck_Label1,0,0);
TAC(TAC_LESS,sPeRWeIRdFck_Temp0,a,b);
TAC(TAC_JFALSE,sPeRWeIRdFck_Label0,sPeRWeIRdFck_Temp0,0);
TAC(TAC_ADD,sPeRWeIRdFck_Temp1,a,1);
TAC(TAC_ASSIGN,a,sPeRWeIRdFck_Temp1,0);
TAC(TAC_PRINT,0,a,0);
TAC(TAC_JMP,sPeRWeIRdFck_Label1,0,0);
TAC(TAC_LABEL,sPeRWeIRdFck_Label0,0,0);
TAC(TAC_ENDFUNC,main,0,0);
```

TACs geradas

```

1  ##FIXED INIT
2  printintstr: > .asciz > "%d\n"
3  printstringst: .asciz "%s\n"
4
5  .text
6  ## TAC_BEGIN_FUN
7  > .globl > main
8  > .type > main, @function
9  main:
10 > pushq > %rbp
11 > movq > %rsp, %rbp
12 ## TAC_PRINT
13 > movl > _a(%rip), %esi
14 > leaq > printintstr(%rip), %rax
15 > movq > %rax, %rdi
16 > call > printf@PLT
17 ## TAC_ASSIGN
18 > movl > _8(%rip), %eax
19 > movl > %eax, _b(%rip)
20 ## TAC_PRINT
21 > movl > _b(%rip), %esi
22 > leaq > printintstr(%rip), %rax
23 > movq > %rax, %rdi
24 > call > printf@PLT
25 ## TAC_LABEL
26 .sPeRWeIRdFck_Label1:
27 ## TAC_LESS
28 > movl > _a(%rip), %edx
29 > movl > _b(%rip), %eax
30 > cmpl > %eax, %edx
31 > setl > %al
32 > movzbl > %al, %eax
33 > movl > %eax, _sPeRWeIRdFck_Temp0(%rip)
34 ## TAC_JFALSE
35 > movl > _sPeRWeIRdFck_Temp0(%rip), %eax
36 > testl > %eax, %eax
37 > je > .sPeRWeIRdFck_Label0
38 ## TAC_ADD
39 > movl > _a(%rip), %edx
40 > movl > _1(%rip), %eax
41 > addl > %edx, %eax
42 > movl > %eax, _sPeRWeIRdFck_Temp1(%rip)
43 ## TAC_ASSIGN
44 > movl > _sPeRWeIRdFck_Temp1(%rip), %eax
45 > movl > %eax, _a(%rip)
46 ## TAC_PRINT
47 > movl > _a(%rip), %esi
48 > leaq > printintstr(%rip), %rax
49 > movq > %rax, %rdi
50 > call > printf@PLT
51 ## TAC_JMP
52 > jmp > .sPeRWeIRdFck_Label1

```

ASM gerado parte 1

```

## TAC_LABEL
.sPeRWeIRDfCk_Label0:
## TAC_END_FUN:
>    popq>    %rbp
>    ret
.LFE0:
>    .size>    main, .-main
>    .ident>   "GCC: (GNU) 12.1.1 20220730"
>    .section> .note.GNU-stack,"",@progbits
## DATA SECTION
>    .section> .rodata

>    .data
_1:> .long> 1
_2:> .long> 2
_8:> .long> 8
_a:> .long> 2
_b:> .long> 777
_c:> .long> 111
_111:> .long> 111
_sPeRWeIRDfCk_Label0:> .long> 0
_sPeRWeIRDfCk_Temp0:> .long> 0
_sPeRWeIRDfCk_Label1:> .long> 0
_777:> .long> 777
_sPeRWeIRDfCk_Temp1:> .long> 0

```

ASM gerado parte 2

```

~ / Ricardo / UFRGS / compiladores / ASM  P main !15 ? gcc out.s
~ / Ricardo / UFRGS / compiladores / ASM  P main !16 ? ./a.out
2
8
3
4
5
6
7
8
~ / Ricardo / UFRGS / compiladores / ASM  P main !16 ?

```

Compilação GCC e execução

2.

```

1
2   int a(2);
3   int b(777);
4   int c(111);
5
6
7   int main (){
8
9       >>   if(a>b){
10          >>       a<-5;
11          >>       print(a);}
12          >>   else{
13              >>       b<-7;
14              >>       print(b);
15          }
16
17      }

```

Código 2

No código dois temos a implementação do comando IF THEN ELSE, onde as variáveis a e b são comparadas e como a não é maior que b, o comando executa o else, em que assinala 7 em b e imprime ele.

```

Table[23] has main of type 5 and datatype 2
Table[50] has 2 of type 2 and datatype 0
Table[53] has 5 of type 2 and datatype 0
Table[55] has 7 of type 2 and datatype 0
Table[97] has a of type 6 and datatype 2
Table[98] has b of type 6 and datatype 2
Table[99] has c of type 6 and datatype 2
Table[459] has 111 of type 2 and datatype 0
Table[962] has 777 of type 2 and datatype 0
TAC(TAC_BEGINFUNC,main,0,0);
TAC(TAC_BIG,sPerWeIRDfCk_Temp0,a,b);
TAC(TAC_JFALSE,sPerWeIRDfCk_Label0,sPerWeIRDfCk_Temp0,0);
TAC(TAC_ASSIGN,a,5,0);
TAC(TAC_PRINT,0,a,0);
TAC(TAC_JMP,sPerWeIRDfCk_Label1,0,0);
TAC(TAC_LABEL,sPerWeIRDfCk_Label0,0,0);
TAC(TAC_ASSIGN,b,7,0);
TAC(TAC_PRINT,0,b,0);
TAC(TAC_LABEL,sPerWeIRDfCk_Label1,0,0);
TAC(TAC_ENDFUNC,main,0,0);

```

TACs geradas

```

1  ##FIXED INIT
2  printintstr:> .asciz> "%d\n"
3  printstringst: .asciz "%s\n"
4
5  .text
6  ## TAC_BEGIN_FUN
7  >> .globl> main
8  >> .type> main, @function
9  main:
10 >> pushq> %rbp
11 >> movq> %rsp, %rbp
12 ## TAC_BIG
13 >> movl> _a(%rip), %edx
14 >> movl> _b(%rip), %eax
15 >> cmpl> %eax,%edx
16 >> setg> %al
17 >> movzbl> %al,%eax
18 >> movl> %eax, _sPeRWeIRDfCk_Temp0(%rip)
19 ## TAC_JFALSE
20 >> movl> _sPeRWeIRDfCk_Temp0(%rip), %eax
21 >> testl> %eax, %eax
22 >> je> .sPeRWeIRDfCk_Label0
23 ## TAC_ASSIGN
24 >> movl> _5(%rip), %eax
25 >> movl> %eax, _a(%rip)
26 ## TAC_PRINT
27 >> movl> _a(%rip), %esi
28 >> leaq> printintstr(%rip), %rax
29 >> movq> %rax, %rdi
30 >> call> printf@PLT
31 ## TAC_JMP
32 >> jmp> .sPeRWeIRDfCk_Label1
33 ## TAC_LABEL
34 .sPeRWeIRDfCk_Label0:
35 ## TAC_ASSIGN
36 >> movl> _7(%rip), %eax
37 >> movl> %eax, _b(%rip)
38 ## TAC_PRINT
39 >> movl> _b(%rip), %esi
40 >> leaq> printintstr(%rip), %rax
41 >> movq> %rax, %rdi
42 >> call> printf@PLT
43 ## TAC_LABEL
44 .sPeRWeIRDfCk_Label1:
45 ## TAC_END_FUN:
46 >> popq> %rbp
47 >> ret
48 .LFE0:
49 >> .size> main, .-main
50 >> .ident> "GCC: (GNU) 12.1.1 20220730"
51 >> .section> .note.GNU-stack,"",@progbits
52 ## DATA SECTION

```

ASM Gerado parte 1

```

53  >> .section> .rodata
54
55  >> .data
56  _2:>>.long> 2
57  _5:>>.long> 5
58  _7:>>.long> 7
59  _a:>>.long> 2
60  _b:>>.long> 777
61  _c:>>.long> 111
62  _111:>> .long> 111
63  _sPeRWeIRdFck_Label0:>> .long> 0
64  _sPeRWeIRdFck_Temp0:>> .long> 0
65  _sPeRWeIRdFck_Label1:>> .long> 0
66  _777:>> .long> 777
67

```

ASM Gerado parte 2

```

~/Ricardo/UFRGS/compiladores/ASM  P main !1 gcc out.s
~/Ricardo/UFRGS/compiladores/ASM  P main !1 ./a.out
7
~/Ricardo/UFRGS/compiladores/ASM  P main !1

```

Compilação e execução