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

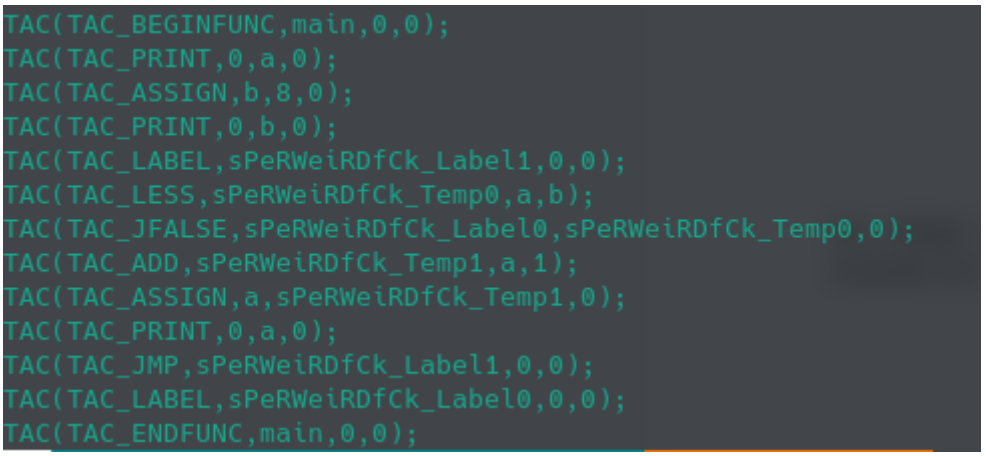
Testes

1.

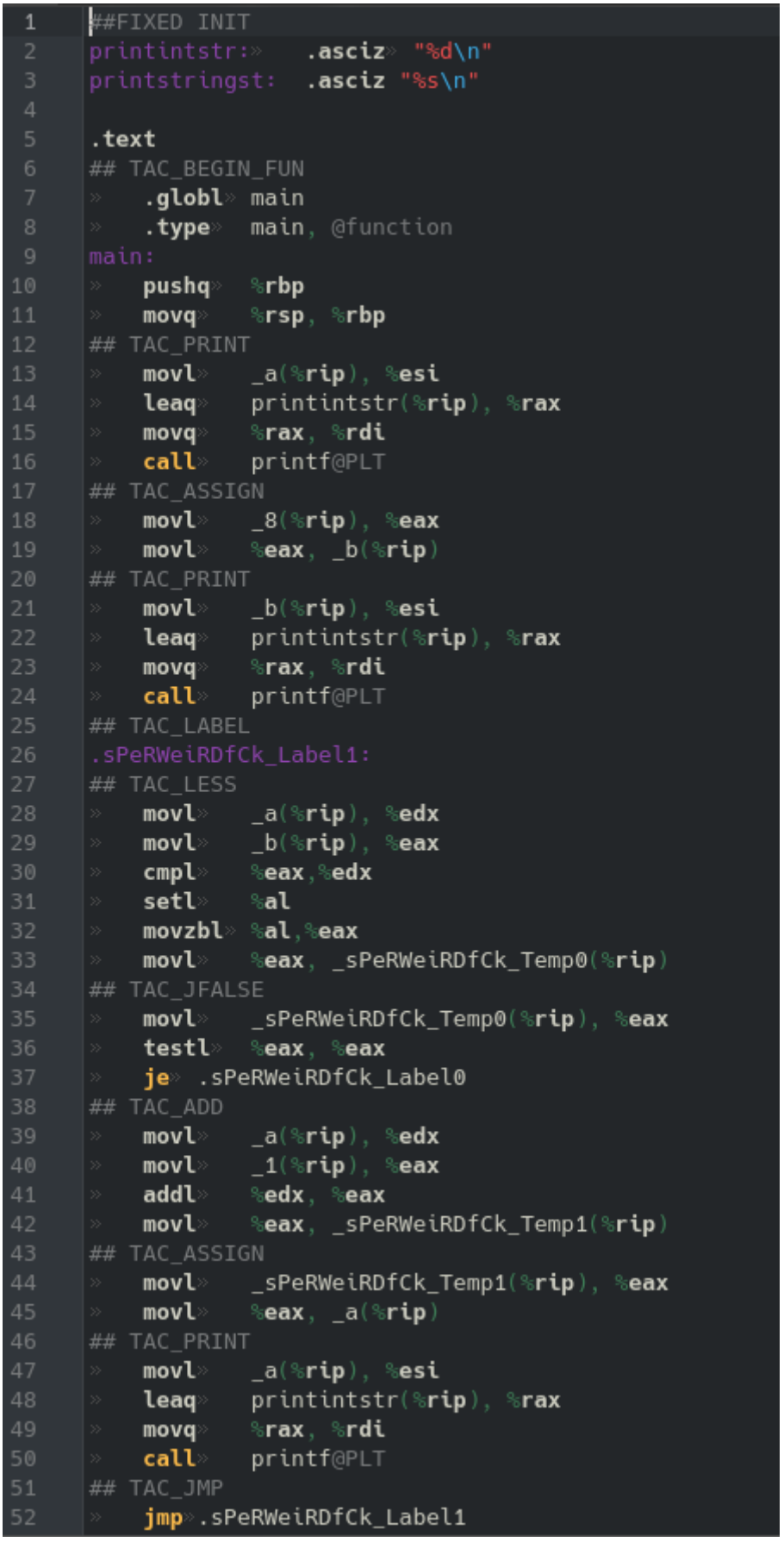
```
1
2  int a(2);
3  int b(777);
4  int c(111);
5
6
7  int main (){
8
9  »   print(a);
10 »   b <- 8;
11 »   print(b);
12 »   while(a<b){
13 »   »   a <- a+1;
14 »   »   print(a);
15 »   };
16
17 }
18
19
```

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.



TACs geradas



```
##FIXED INIT
printintstr:    .asciz "%d\n"
printstringst:  .asciz "%s\n"

.text
## TAC_BEGIN_FUN
    .globl main
    .type  main, @function
main:
    pushq  %rbp
    movq   %rsp, %rbp
## TAC_PRINT
    movl   _a(%rip), %esi
    leaq   printintstr(%rip), %rax
    movq   %rax, %rdi
    call   printf@PLT
## TAC_ASSIGN
    movl   _8(%rip), %eax
    movl   %eax, _b(%rip)
## TAC_PRINT
    movl   _b(%rip), %esi
    leaq   printintstr(%rip), %rax
    movq   %rax, %rdi
    call   printf@PLT
## TAC_LABEL
.sPeRWeiRDfCk_Label1:
## TAC_LESS
    movl   _a(%rip), %edx
    movl   _b(%rip), %eax
    cmpl   %eax,%edx
    setl   %al
    movzbl %al,%eax
    movl   %eax, _sPeRWeiRDfCk_Temp0(%rip)
## TAC_JFALSE
    movl   _sPeRWeiRDfCk_Temp0(%rip), %eax
    testl  %eax, %eax
    je  .sPeRWeiRDfCk_Label0
## TAC_ADD
    movl   _a(%rip), %edx
    movl   _1(%rip), %eax
    addl   %edx, %eax
    movl   %eax, _sPeRWeiRDfCk_Temp1(%rip)
## TAC_ASSIGN
    movl   _sPeRWeiRDfCk_Temp1(%rip), %eax
    movl   %eax, _a(%rip)
## TAC_PRINT
    movl   _a(%rip), %esi
    leaq   printintstr(%rip), %rax
    movq   %rax, %rdi
    call   printf@PLT
## TAC_JMP
    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  main !15 ? gcc out.s
~/Ricardo/UFRGS/compiladores/ASM  main !16 ? ./a.out
2
8
3
4
5
6
7
8
~/Ricardo/UFRGS/compiladores/ASM  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

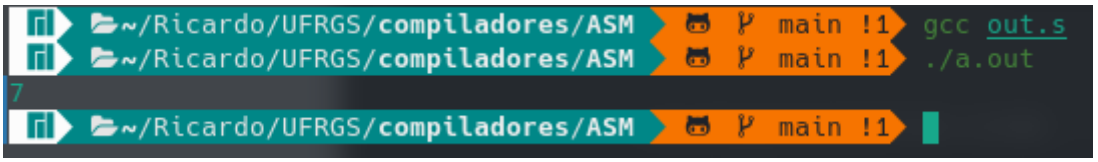
```
##FIXED INIT
printintstr:    .asciz "%d\n"
printstringst:  .asciz "%s\n"

.text
## TAC_BEGIN_FUN
    .globl main
    .type  main, @function
main:
    pushq  %rbp
    movq   %rsp, %rbp
## TAC_BIG
    movl   _a(%rip), %edx
    movl   _b(%rip), %eax
    cmpl   %eax,%edx
    setg   %al
    movzbl %al,%eax
    movl   %eax, _sPeRWeiRDfCk_Temp0(%rip)
## TAC_JFALSE
    movl   _sPeRWeiRDfCk_Temp0(%rip), %eax
    testl  %eax, %eax
    je  .sPeRWeiRDfCk_Label0
## TAC_ASSIGN
    movl   _5(%rip), %eax
    movl   %eax, _a(%rip)
## TAC_PRINT
    movl   _a(%rip), %esi
    leaq   printintstr(%rip), %rax
    movq   %rax, %rdi
    call   printf@PLT
## TAC_JMP
    jmp .sPeRWeiRDfCk_Label1
## TAC_LABEL
.sPeRWeiRDfCk_Label0:
## TAC_ASSIGN
    movl   _7(%rip), %eax
    movl   %eax, _b(%rip)
## TAC_PRINT
    movl   _b(%rip), %esi
    leaq   printintstr(%rip), %rax
    movq   %rax, %rdi
    call   printf@PLT
## TAC_LABEL
.sPeRWeiRDfCk_Label1:
## TAC_END_FUN:
    popq   %rbp
    ret
.LFE0:
    .size  main, .-main
    .ident "GCC: (GNU) 12.1.1 20220730"
    .section    .note.GNU-stack,"",@progbits
## 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



Compilação e execução