Identificacão:

Nome: Augusto Zanella Bardini (00278083)

Trabalho de Compiladores, etapa 6: Geração de ASM.

Informações básicas

> Para executar o programa fim a fim em minha máquina, utilizei:

```
make clean; make && ./etapa6 testeFibo.azby out.txt
```

A saida deve mostrar na stderr:

- AST
- TACs
- HASH Table

> Para ver o ASM gerado, com comentários por TAC (iguais aos exemplos):

```
cat out.s
```

> Para executar o ASM gerado (não sei se funciona em Mac pois minha arquitetura não permite a main começando por underline)

```
gcc out.s && ./a.out
```

O que/como foi feito:

Vou descrever resumidamente as principais alterações do código abaixo:

Alterações nas TACs

- Gerada uma nova TAC para lidar com atribuições de vetor para vetor, pois uma só não conseguia contemplar todos os nomes de variáveis e valores.
- Alterada a tac parametros de função para fazer match entre o índice de cada parametro e a declaração, percorrendo recursivamente a AST
- Corrigida TAC de `while` que não continha um label final

Geração de ASM

- Assim como mostrado na aula, para cada TAC, busquei reproduzir um código C que as representasse, entender o funcionamento, e "clonar" as instruções geradas de forma dinâmica para a linguagem da disciplina.
- Foi necessario adicionar os valores de inicialização de variaveis e vetores na tabela hash, fiz no formato de ponteiros pra AST, e percorro recursivamente.
- Alguns pequenos exemplos de código C ainda restaram na pasta `c_sample`, deixei eles lá para que possa set visto e comparado brevemente.

O que fiquei devendo:

- Operações com float;
- Declarações sem inicialização;
- Algumas formatações de print;

Exemplos:

Exemplo 1:

No seguinte programa, é executado um simples Fibbonacci:

```
int i: 2;
int fib_um: 0;
int fib_dois: 1;
int res: 0;

int main(){
    while (i <= 10) {
        i = i+1;
        res = fib_um + fib_dois;
        fib_um = fib_dois;
        fib_dois = res;
        print res;
    };
}
```

O resultado da execução mostra a sequencia de Fibbonacci:

```
1
2
3
5
8
13
21
34
55
```

O ASM gerado pode ser visto abaixo:

```
.data
_0:
   .long    0
_1:
   .long    1
_2:
   .long    2
_i:
   .long    2
_res:
   .long    0
_10:
   .long    10
_OMEGATemp0:
   .long    0
_OMEGATemp1:
   .long    0
_OMEGATemp2:
   .long    0
_fib_um:
   .long    0
_fib_dois:
   .long    1

.formatInt:
   .string  "%d\n"
.formatString:
   .string  "%s\n"
.formatChar:
   .string  "%c\n"
#-------------------------TAC_FUN_START
.text
.globl  main
main:
   pushq    %rbp
   movq %rsp, %rbp

#-------------------------TAC_LABEL
.LAMBDALabel0:
```

```
#-------------------------TAC_LEQ
movq %rsp, %rbp
movl _i(%rip), %edx
movl _10(%rip), %eax
cmpl %eax, %edx
setle    %al
movzbl   %al, %eax
movl %eax, _OMEGATemp0(%rip)

#-------------------------TAC_JUMP_FALSE
movl _OMEGATemp0(%rip), %eax
testl  %eax, %eax
je   .LAMBDALabel2

#-------------------------TAC_SUM
movl _i(%rip), %edx
movl _1(%rip), %eax
addl %edx, %eax
movl %eax, _OMEGATemp1(%rip)

#-------------------------TAC_ATTR
movl _OMEGATemp1(%rip), %eax
movl %eax, _i(%rip)

#-------------------------TAC_SUM
movl _fib_um(%rip), %edx
movl _fib_dois(%rip), %eax
addl %edx, %eax
movl %eax, _OMEGATemp2(%rip)

#-------------------------TAC_ATTR
movl _OMEGATemp2(%rip), %eax
movl %eax, _res(%rip)

#-------------------------TAC_ATTR
movl _fib_dois(%rip), %eax
movl %eax, _fib_um(%rip)

#-------------------------TAC_ATTR
movl _res(%rip), %eax
movl %eax, _fib_dois(%rip)

#-------------------------TAC_PRNT:INT
movl _res(%rip), %eax
movl %eax, %esi
leaq .formatInt(%rip), %rdi
movl $0, %eax
call printf@PLT

#-------------------------TAC_JUMP
jmp  .LAMBDALabel0
```

```
#-------------------------TAC_LABEL
.LAMBDALabel2:

   #-------------------------TAC_FUN_END
   popq %rbp
   ret
```

Exemplo 2:

No seguinte programa, dado pelo professor, comentei as linhas que não conseguem ser compiladas e executei o programa.

```
\* Abaixo arquivo exemplo fornecido pelo professor: *\
\* UFRGS - Compiladores - Marcelo Johann - 2021/2 *\

char c: 'x';
char d: 100;
int a: 'A';
int i: 1;
int v[10]: 'a' 0 0 0 0 0 0 0 0 0;
\\ int matrix[100]; #COMENTADO
float f: 2/3;

\*
This is a comment
of multiple lines
*\

int main ()
{
    a = 0;
    a = a - i;
    a = 5;
    v[a] = 55;
    \\ print v[5]; #COMENTADO
    print a;
    i = 2;

    print "Digite um numero: \n";
    a = read;

    while i<10
    {
        i = incn(i,1);
        a = incn(a,1);
    };
```

```
    print "Incrementado algumas vezes a fica " , a ,"\n";

    if a==15 then
    {
        label-x:
        a = a - 1;
        print "A era=15\n";
    };

    if (i==100) then
    {
        print "Nao tem como isso...\n";
    }
    else
        print "OK!\n";
    if a > 0 then
        goto label-x;
}

int incn (int x , int n)
{
    return x+n;
}``
```

O resultado das execuções estão abaixo, com entradas `-5` e `0`.

```
5
Digite um numero:

-5\ #MINHA ENTRADA
Incrementado algumas vezes a fica
3


OK!

A era=15

OK!

A era=15

OK!

A era=15

OK!
```

```
5
Digite um numero:

0\n #MINHA ENTRADA
Incrementado algumas vezes a fica
8


OK!

A era=15

OK!

A era=15

OK!

A era=15

OK!

A era=15

OK!

A era=15

OK!

A era=15

OK!

A era=15

OK!

A era=15

OK!

A era=15

OK!
```

O ASM gerado para esse código pode ser visto abaixo:

```
.data
_0:
   .long    0
_1:
   .long    1
_2:
   .long    2
```

```
_3:
   .long    3
_5:
   .long    5
_a:
   .long    'A'
_c:
   .long    'x'
_d:
   .long    100
_f:
   .long    2
_i:
   .long    1
_n:
   .long    0
_v:
   .long    'a'
   .long    0
   .long    0
   .long    0
   .long    0
   .long    0
   .long    0
   .long    0
   .long    0
   .long    0
_x:
   .long    0
._Incrementado_algumas_vezes_a_fica_:
   .string    "Incrementado algumas vezes a fica \n"
_100:
   .long    100
_OMEGATemp5:
   .long    0
_OMEGATemp6:
   .long    0
._OK__n:
   .string    "OK!\n\n"
_OMEGATemp7:
   .long    0
_97:
   .byte    97
_65:
   .byte    65
_OMEGATemp8:
   .long    0
_OMEGATemp9:
   .long    0
_120:
   .byte    120
_10:
```

```
   .long    10
_15:
   .long    15
_55:
   .long    55
_OMEGATemp0:
   .long    0
_OMEGATemp1:
   .long    0
.__n:
   .string    "\n\n"
._Digite_um_numero___n:
   .string    "Digite um numero: \n\n"
._A_era____n:
   .string    "A era=15\n\n"
_OMEGATemp2:
   .long    0
_OMEGATemp3:
   .long    0
._Nao_tem_como_isso____n:
   .string    "Nao tem como isso...\n\n"
_OMEGATemp4:
   .long    0

.formatInt:
   .string  "%d\n"
.formatString:
   .string  "%s\n"
.formatChar:
   .string  "%c\n"
#-------------------------TAC_FUN_START
.text
.globl  main
main:
   pushq    %rbp
   movq %rsp, %rbp

   #-------------------------TAC_ATTR
   movl _0(%rip), %eax
   movl %eax, _a(%rip)

   #-------------------------TAC_SUB
   movl _a(%rip), %edx
   movl _i(%rip), %eax
   subl %eax, %edx
   movl %edx, %eax
   movl %eax, _OMEGATemp0(%rip)

   #-------------------------TAC_ATTR
   movl _OMEGATemp0(%rip), %eax
   movl %eax, _a(%rip)
```

```
    #-------------------------TAC_ATTR
    movl _5(%rip), %eax
    movl %eax, _a(%rip)

    #-------------------------TAC_ATTR_VEC_VAL
    movl _55(%rip), %eax

    #-------------------------TAC_ATTR_VEC
    movl %eax, 0+_v(%rip)

    #-------------------------TAC_PRNT:INT
    movl _5(%rip), %eax
    movl %eax, %esi
    leaq .formatInt(%rip), %rdi
    movl $0, %eax
    call printf@PLT

    #-------------------------TAC_PRNT:INT
    movl _a(%rip), %eax
    movl %eax, %esi
    leaq .formatInt(%rip), %rdi
    movl $0, %eax
    call printf@PLT

    #-------------------------TAC_ATTR
    movl _2(%rip), %eax
    movl %eax, _i(%rip)

    #-------------------------TAC_PRNT:STRING
    leaq ._Digite_um_numero___n(%rip), %rdi
    movl $0, %eax
    call printf@PLT

    #-------------------------TAC_READ
    leaq _OMEGATemp1(%rip), %rsi
    leaq .formatInt(%rip), %rdi
    movl $0, %eax
    call __isoc99_scanf@PLT

    #-------------------------TAC_ATTR
    movl _OMEGATemp1(%rip), %eax
    movl %eax, _a(%rip)

#-------------------------TAC_LABEL
.LAMBDALabel0:

    #-------------------------TAC_LES
    movq %rsp, %rbp
    movl _i(%rip), %edx
    movl _10(%rip), %eax
    cmpl %eax, %edx
    setl %al
```

```
    movzbl   %al, %eax
    movl %eax, _OMEGATemp2(%rip)

    #-------------------------TAC_JUMP_FALSE
    movl _OMEGATemp2(%rip), %eax
    testl  %eax, %eax
    je   .LAMBDALabel2

    #-------------------------TAC_PARAM_CALL
    movl _1(%rip), %eax
    movl %eax, _n(%rip)

    #-------------------------TAC_PARAM_CALL
    movl _i(%rip), %eax
    movl %eax, _x(%rip)

    #-------------------------TAC_FUN_CALL
    call _incn
    movl %eax, _OMEGATemp3(%rip)

    #-------------------------TAC_ATTR
    movl _OMEGATemp3(%rip), %eax
    movl %eax, _i(%rip)

    #-------------------------TAC_PARAM_CALL
    movl _1(%rip), %eax
    movl %eax, _n(%rip)

    #-------------------------TAC_PARAM_CALL
    movl _a(%rip), %eax
    movl %eax, _x(%rip)

    #-------------------------TAC_FUN_CALL
    call _incn
    movl %eax, _OMEGATemp4(%rip)

    #-------------------------TAC_ATTR
    movl _OMEGATemp4(%rip), %eax
    movl %eax, _a(%rip)

    #-------------------------TAC_JUMP
    jmp  .LAMBDALabel0

#-------------------------TAC_LABEL
.LAMBDALabel2:

    #-------------------------TAC_PRNT:STRING
    leaq ._Incrementado_algumas_vezes_a_fica_(%rip), %rdi
    movl $0, %eax
    call printf@PLT

    #-------------------------TAC_PRNT:INT
```

```
    movl _a(%rip), %eax
    movl %eax, %esi
    leaq .formatInt(%rip), %rdi
    movl $0, %eax
    call printf@PLT

    #-------------------------TAC_PRNT:STRING
    leaq .__n(%rip), %rdi
    movl $0, %eax
    call printf@PLT

    #-------------------------TAC_EQU
    movq %rsp, %rbp
    movl _a(%rip), %edx
    movl _15(%rip), %eax
    cmpl %eax, %edx
    sete %al
    movzbl   %al, %eax
    movl %eax, _OMEGATemp5(%rip)

    #-------------------------TAC_JUMP_FALSE
    movl _OMEGATemp5(%rip), %eax
    testl  %eax, %eax
    je   .LAMBDALabel3

#-------------------------TAC_LABEL
.label_x:

    #-------------------------TAC_SUB
    movl _a(%rip), %edx
    movl _1(%rip), %eax
    subl %eax, %edx
    movl %edx, %eax
    movl %eax, _OMEGATemp6(%rip)

    #-------------------------TAC_ATTR
    movl _OMEGATemp6(%rip), %eax
    movl %eax, _a(%rip)

    #-------------------------TAC_PRNT:STRING
    leaq ._A_era____n(%rip), %rdi
    movl $0, %eax
    call printf@PLT

#-------------------------TAC_LABEL
.LAMBDALabel3:

    #-------------------------TAC_EQU
    movq %rsp, %rbp
    movl _i(%rip), %edx
    movl _100(%rip), %eax
    cmpl %eax, %edx
```

```
    sete %al
    movzbl   %al, %eax
    movl %eax, _OMEGATemp7(%rip)

    #-------------------------TAC_JUMP_FALSE
    movl _OMEGATemp7(%rip), %eax
    testl  %eax, %eax
    je   .LAMBDALabel4

    #-------------------------TAC_PRNT:STRING
    leaq ._Nao_tem_como_isso____n(%rip), %rdi
    movl $0, %eax
    call printf@PLT

    #-------------------------TAC_JUMP
    jmp  .LAMBDALabel5

#-------------------------TAC_LABEL
.LAMBDALabel4:

    #-------------------------TAC_PRNT:STRING
    leaq ._OK__n(%rip), %rdi
    movl $0, %eax
    call printf@PLT

#-------------------------TAC_LABEL
.LAMBDALabel5:

    #-------------------------TAC_GRE
    movq %rsp, %rbp
    movl _a(%rip), %edx
    movl _0(%rip), %eax
    cmpl %eax, %edx
    setg %al
    movzbl   %al, %eax
    movl %eax, _OMEGATemp8(%rip)

    #-------------------------TAC_JUMP_FALSE
    movl _OMEGATemp8(%rip), %eax
    testl  %eax, %eax
    je   .LAMBDALabel6

    #-------------------------TAC_GOTO
    jmp  .label_x

#-------------------------TAC_LABEL
.LAMBDALabel6:

    #-------------------------TAC_FUN_END
    popq %rbp
    ret
```

```
#-------------------------TAC_FUN_START
.text
.globl  _incn
_incn:
   pushq    %rbp
   movq %rsp, %rbp

   #-------------------------TAC_SUM
   movl _x(%rip), %edx
   movl _n(%rip), %eax
   addl %edx, %eax
   movl %eax, _OMEGATemp9(%rip)

   #-------------------------TAC_RTRN
   movl _OMEGATemp9(%rip), %eax
   popq %rbp
   ret

   #-------------------------TAC_FUN_END
   popq %rbp
   ret
```

Exemplo 3

O seguinte código faz operações diversas e da jump para dentro de um if impossível:

```
int a: 0;
int b: 20;
int c: 1;

int main () {
    a = read;
    if (0 != 0) then {
        never:
        a = 20;
    };
    while (a < 10) {
        a = a + 1;
        print a;
    };
    if a == 10 then
        goto never;
    print a;
}
```

O resultado da execução:

```
5\n #MINHA ENTRADA
6
7
8
9
10
20
```

O ASM gerado:

```
.data
_0:
   .long    0
_1:
   .long    1
_a:
   .long    0
_b:
   .long    20
_c:
   .long    1
_10:
   .long    10
_20:
   .long    20
_OMEGATemp0:
   .long    0
_OMEGATemp1:
   .long    0
_OMEGATemp2:
   .long    0
_OMEGATemp3:
   .long    0
_OMEGATemp4:
   .long    0

.formatInt:
   .string  "%d\n"
.formatString:
   .string  "%s\n"
.formatChar:
   .string  "%c\n"
#-------------------------TAC_FUN_START
.text
.globl  main
main:
   pushq    %rbp
   movq %rsp, %rbp

   #-------------------------TAC_READ
   leaq _OMEGATemp0(%rip), %rsi
```

```
    leaq .formatInt(%rip), %rdi
    movl $0, %eax
    call __isoc99_scanf@PLT

    #-------------------------TAC_ATTR
    movl _OMEGATemp0(%rip), %eax
    movl %eax, _a(%rip)

    #-------------------------TAC_DIF
    movq %rsp, %rbp
    movl _0(%rip), %edx
    movl _0(%rip), %eax
    cmpl %eax, %edx
    setne    %al
    movzbl   %al, %eax
    movl %eax, _OMEGATemp1(%rip)

    #-------------------------TAC_JUMP_FALSE
    movl _OMEGATemp1(%rip), %eax
    testl  %eax, %eax
    je   .LAMBDALabel0

#-------------------------TAC_LABEL
.never:

    #-------------------------TAC_ATTR
    movl _20(%rip), %eax
    movl %eax, _a(%rip)

#-------------------------TAC_LABEL
.LAMBDALabel0:

#-------------------------TAC_LABEL
.LAMBDALabel1:

    #-------------------------TAC_LES
    movq %rsp, %rbp
    movl _a(%rip), %edx
    movl _10(%rip), %eax
    cmpl %eax, %edx
    setl %al
    movzbl   %al, %eax
    movl %eax, _OMEGATemp2(%rip)

    #-------------------------TAC_JUMP_FALSE
    movl _OMEGATemp2(%rip), %eax
    testl  %eax, %eax
    je   .LAMBDALabel3

    #-------------------------TAC_SUM
    movl _a(%rip), %edx
    movl _1(%rip), %eax
```

```
    addl %edx, %eax
    movl %eax, _OMEGATemp3(%rip)

    #-------------------------TAC_ATTR
    movl _OMEGATemp3(%rip), %eax
    movl %eax, _a(%rip)

    #-------------------------TAC_PRNT:INT
    movl _a(%rip), %eax
    movl %eax, %esi
    leaq .formatInt(%rip), %rdi
    movl $0, %eax
    call printf@PLT

    #-------------------------TAC_JUMP
    jmp  .LAMBDALabel1

#-------------------------TAC_LABEL
.LAMBDALabel3:

    #-------------------------TAC_EQU
    movq %rsp, %rbp
    movl _a(%rip), %edx
    movl _10(%rip), %eax
    cmpl %eax, %edx
    sete %al
    movzbl   %al, %eax
    movl %eax, _OMEGATemp4(%rip)

    #-------------------------TAC_JUMP_FALSE
    movl _OMEGATemp4(%rip), %eax
    testl  %eax, %eax
    je   .LAMBDALabel4

    #-------------------------TAC_GOTO
    jmp  .never

#-------------------------TAC_LABEL
.LAMBDALabel4:

    #-------------------------TAC_PRNT:INT
    movl _a(%rip), %eax
    movl %eax, %esi
    leaq .formatInt(%rip), %rdi
    movl $0, %eax
    call printf@PLT

    #-------------------------TAC_FUN_END
    popq %rbp
    ret
```