PCS2056 – Linguagens e Compiladores

5ª parte do projeto – Tradução de comandos

Renan Martins Zomignani Mendes
Tiago Schelp Lopes

Introdução

Esse documento visa apresentar algumas traduções de comandos como serão implementadas pelo compilador.

Além de estruturas de controle de fluxo como if e while será apresentado a tradução de controles comandos imperativos e a abordagem de chamadas de subrotina.

Tradução de estruturas de controle de fluxo:

Desvio

Nossa linguagem

Desvio -> nao implementado

Linguagem de saída

JP endereço_destino

If-then

Nossa linguagem

```
If(condicao)
        comandos
end
```

Linguagem de saída

	;(expressão)
	JZ End
	;(comandos)
End	

If-then-else

Nossa linguagem:

```
If(condicao)
        comandos
elsif(condicao)
          comandos
end
```

Linguagem de saída:

	;(expressão)
	JZ ELSE
	;(comandos)
	JP FIM
ELSE	OS /0
	;(comandos)
FIM	OS /0

While

Nossa linguagem:

```
while(condicao)
        comandos
end
```

Linguagem de saída:

WHILE	OS /0
	;(expressão)
	JZ End
	;(comandos)
	JP WHILE
End	OS /0

Tradução de comandos imperativos

Atribuição de valor

Nossa linguagem:

```
a = 4
```

Linguagem de saída:

	LD /0004
	MM a

"a" deve estar definido:

```
a K /0000
```

Leitura (entrada)

Nossa Linguagem:

```
#A leitura é feita de um único caractere numérico
input a
```

Linguagem de saída:

	GD /000
	/ K100
	- k30 ; converte de ASC para decimal
	MM a ;

K30 é constante previamente declarada

```
K30 k /0030
```

Impressão (saída)

Nossa Linguagem:

#imprime um caractere numérico referente ao número armazenado
output a

Linguagem de saída:

	LD a
	+ K30 ; converte para ASC
	PD /100 ; imprime no monitor

K30 é constante previamente declarada de valor = 30
K30 k /0030

Chamada de subrotina

Nossa linguagem:

Y = 2
a = subrotina(1, Y)

Linguagem de saída:

Área de programa

	LV /0002
	MM Y
	LV /0001
	MM subrotina_p1
	LD Y
	MM subrotina_p2
	SC SUBROTINA
	LD retorno
	MM a

Subrotina(Param1,Param2)

SUBROTINA	OS /0
	; (comandos)
	MM retorno ; armazena resultado
	RS SUBROTINA

Área de variáveis

subrotina_p1	K /0000
subrotina_p2	K /0000
Y	K /0000

retorno	K /0000
a	K /0000

“retorno” é uma variável usada para retorno de funções

Essa tradução é uma simplificação do modelo teórico que implementa registros de ativação.
Da maneira como foi traduzido, é possível que uma função chame outras funções, sem perder a referência do endereço de retorno nem misturar as variáveis em cada escopo.

Modelo implementado

Da maneira como foi apresentado uma função pode chamar outra função, evitando que o conteúdo das variáveis sejam sobrepostos ou que os endereços de retorno de cada função sejam perdidos.

Os endereços de memória das variáveis de cada subrotina são únicos, mesmo que o nome das variáveis na linguagem que definimos sejam as mesmas, em tempo de compilação o mecanismo pensado cria um label para criar essa diferenciação, nesse momento o compilador adiciona um prefixo no nome da variável com o nome da subrotina de modo a obter: NOMESUBROTINA_NOMEVARIAVEL.

Uma limitação desse modo de implementação é o fato de não ser possível o uso de chamdas recursivas.

Modelo teórico

O registro de ativação é utilizado para definir escopos e evitar que determinados escopos acessem variáveis, dados e temporários de outros escopos bem como garantir que os endereços de retorno de cada função sejam guardados.

Dessa maneira é possível utilizar recursão.

O registro de ativação contém as variáveis e temporários da parte principal do programa, o programa propriamente dito e na ocasião da chamada de subrotina deve empilhas variáveis locais, temporários, StackPointer e endereço de retorno. Esse registro de ativação será acessado apenas quando a subrotina terminar a execução e chamar instrução de retorno.

Todo registro de ativação possui um Stack Pointer que indica o endereço de início do escopo. Esse Stack pointer é atualizado sempre que uma subrotina é chamada, para o início do novo registro de ativação do escopo recém criado, logo acima do último valor empilhado.

Exemplo de programa traduzido

```
#Programa
int main()
        int resultado, a
         input a
        resultado = fatorial(a)
        output resultado
end

#Sub Programa
int fatorial(int n)
        int fat
        fat = 1
                while(n > 0)
                        fat = fat * n
                        n=n-1
                end
        return fatorial
end

@ /0000
JP inicio
.**********************************
;
;area de variaveis
.**********************************
;
resultado       K /000
a               K /000
retorno         K /000
fatorial_n      K /000
fatorial_fat    K /000


.**********************************
;
;area de dados
.**********************************
;
K30             K /030
K100            K /100
K1              K /001
K2              K /002
```

```
;**********************************
; programa
;**********************************
inicio      OS /0
            GD /000     ;le do teclado
            /  K100
            -  K30      ; converte numeros lido
            MM a        ; input a
            LD a
            MM fatorial_n
            SC fatorial ;fatorial a
            LD retorno
            MM resultado
            LD resultado ; le retorno da funcao
            + K30       ; converte para ASC
            PD /100     ; imprime no monitor
            HM /0


;**********************************
;Subrotina fatorial

;**********************************

fatorial    OS /0
            LD K1
            MM fatorial_fat     ; fat = 1
WHILE               OS /0
            LD fatorial_n
            JN FIMWHILE     ;  sai do whole se fatorial_n <= 0
            JZ FIMWHILE
            LD fatorial_fat     ; fat = fat * n
            *  fatorial_n
            MM fatorial_fat
            LD fatorial_n       ; n = n-1
            -  K1
            MM fatorial_n
            JP WHILE
FIMWHILE    OS /0
            LD fatorial_fat
            MM retorno          ; return fat
            RS fatorial
# inicio
```