PCS2056 – Linguagens e Compiladores

5ª parte do projeto

Tradução de comandos

Alan Raso – 804151

Esta parte da entrega é relativa à tradução de comandos da linguagem criada para ser compilada para a linguagem da MVN. Cada capítulo descreve uma estrutura e seu respectivo código traduzido.

## Condição

O comando de condição simples da linguagem objeto é da seguinte forma:

|  |  |
| --- | --- |
| Comando descrito: | Exemplo: |
| se (EXPRESSÃO) {  // COMANDOS\_TRUE  }  // COMANDOS\_FALSE | se (x > 6) {  x := 2;  }  X := -1; |

Para esta estrutura, a expressão que vem após o comando “se” apresentará, após a resolução, um valor numérico. Assim, a condição será interpretada como falsa caso este valor seja 0, e como verdadeira, caso seja diferente de 0. No exemplo acima, se o valor de x for maior do que 6, o valor da expressão avaliada será 1, e caso seja menor do que 6, será 0. A tradução, portanto, é feita da seginte forma:

LD EXPRESSAO\_VALOR

JZ ROT1

<COMANDOS\_TRUE>

ROT1 <COMANDOS\_FALSE>

O comando de condição pode ser acompanhado de “senao” e/ou de outra condição a seguir. Abaixo são apresentadas as traduções para esses casos.

|  |  |
| --- | --- |
| Comando: | Tradução: |
| se (EXPRESSÃO) {  // COMANDOS\_1  } senao {  // COMANDOS\_2  }  // COMANDOS\_3 | LD EXPRESSAO\_VALOR  JZ ROT\_1  <COMANDOS\_1>  ROT\_1 <COMANDOS\_2>  <COMANDOS\_3> |
| se (EXPRESSÃO\_1) {  // COMANDOS\_1  } senao se (EXPRESSÃO\_2) {  // COMANDOS\_2  }  // COMANDOS\_3 | LD EXPRESSÃO\_1\_VALOR  JZ ROT\_1  <COMANDOS\_1>  ROT\_1 LD EXPRESSÃO\_2\_VALOR  JZ ROT\_2  <COMANDOS\_2>  ROT\_2 <COMANDOS\_3> |
| se (EXPRESSÃO\_1) {  // COMANDOS\_1  } senao se (EXPRESSÃO\_2) {  // COMANDOS\_2  } senao {  // COMANDOS\_3  }  // COMANDOS\_4 | LD EXPRESSÃO\_1\_VALOR  JZ ROT\_1  <COMANDOS\_1>  ROT\_1 LD EXPRESSÃO\_2\_VALOR  JZ ROT\_2  <COMANDOS\_2>  ROT\_2 <COMANDOS\_3>  <COMANDOS\_4> |

## Iteração

O comando de iteração traduzido será o “enquanto”, equivalente ao *while*. Novamente existe a avaliação de uma expressão que retorna um valor numérico. Caso este valor seja 0, não se executa o código dentro do bloco. Avalia-se o valor desta expressão a cada final de execução do bloco, ou seja, o que está representado por “EXPRESSÃO” abaixo gera um “EXPRESSÃO\_VALOR” a cada iteração.

|  |  |
| --- | --- |
| Comando: | Tradução: |
| enquanto (EXPRESSÃO) {  // COMANDOS\_1  }  // COMANDOS\_2 | LOOP LD EXPRESSAO\_VALOR  JZ ROT\_1  <COMANDOS\_1>  JP LOOP  ROT\_1 <COMANDOS\_2> |

## Atribuição

A atribuição de um valor a uma variável se dá carregando o resultado da expressão à direita na posição de memória correspondente à variável indicada na esquerda. A utilização do pseudo-comando “K” indica qual o endereço de memória associado a essa variável para que o valor resultado da expressão seja gravado nesta posição. Para uma variável chamada “num”, portanto, temos a seguinte tradução:

|  |  |
| --- | --- |
| Comando: | Tradução: |
| num := EXPRESSÃO | VAR\_num K /0020  LD EXPRESSAO\_VALOR  MM VAR\_num |

## Leitura (Entrada)

O comando de leitura permite o armazenamento de um valor digitado no teclado para uma variável. Utiliza-se, para a tradução, o comando *GD* da MVN, que guarda o valor digitado no acumulador. É importante ressaltar que o valor armazenado vem em código ASCII, ou seja, deve-se converter esse valor para o armazenamento.

|  |  |
| --- | --- |
| Comando: | Tradução: |
| scanf(“%d”, &num); | VAR\_num K /0020  CTE\_CONV K /0030  GD /0000  - CTE\_CONV  MM VAR\_num |

## Impressão (saída)

Para a impressão aplica-se um processo inverso ao de entrada. Guarda-se o valor contido na posição de memória correspondete no acumulador e utiliza-se o comando PD.

|  |  |
| --- | --- |
| Comando: | Tradução: |
| printf(“%d”, num); | VAR\_num K /0020  CTE\_CONV K /0030  MM VAR\_num  + CTE\_CONV  PD /0000 |

## Chamada de sub-rotina

A chamada de sub-rotina deve utilizar o comando *SC* da MVN para o chamado. Porém, antes da utilização do comando, deve-se guardar os valores dos parâmetros nas posições de memória correspondentes aos argumentos da função. Dado que existem os endereços de memória dos argumentos “termo1” e “termo2” abaixo, e a função “soma” como código da sub-rotina, a tradução é a seguinte:

|  |  |
| --- | --- |
| Comando: | Tradução: |
| func soma(termo1, termo2)(int);  (...)  num = soma(2, 3); | LV /0002  MM termo1  LV /0003  MM termo2  SC soma |