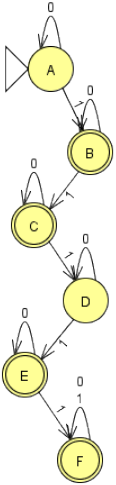


Teoria de Linguagens e Compiladores

Máquina de Turing

Luiz Eduardo da Silva

Universidade Federal de Alfenas

Agenda

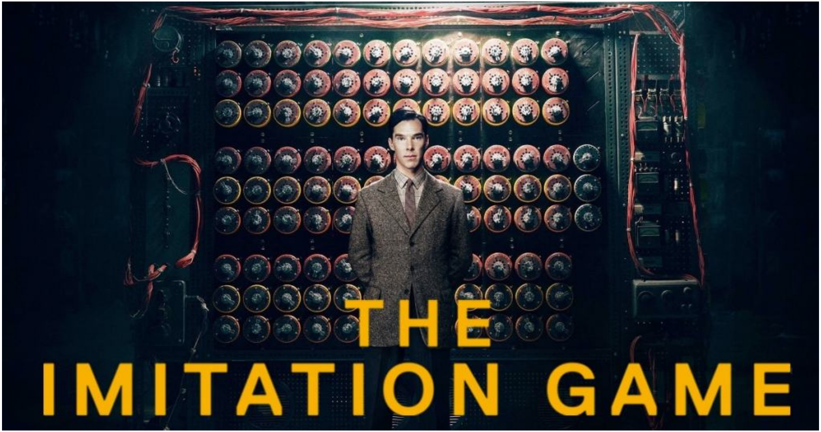
Agenda

1. Máquinas de Turing
   - Introdução
   - Definição Formal
   - Computação da Máquina de Turing
   - Exemplos de Máquinas de Turing
2. Variantes da Máquina de Turing
3. Algoritmos
4. Máquinas Virtuais

Máquina de Turing

- Aprendemos alguns modelos de computação:
  - Autômatos finitos: são dispositivos bons para sistemas de memória limitada e esquemas específicos de computação.
  - Autômato de pilha: estende o potencial do autômato com um modelo de armazenamento do tipo "o último que entra é o primeiro que sai", uma memória ilimitada nesse modelo de computação.
- Ainda assim, algumas tarefas simples não podem ser computadas por esses modelos.
- Apresentamos aqui um modelo mais poderoso, proposto por Alan Turing em 1936, chamado **máquina de Turing**.

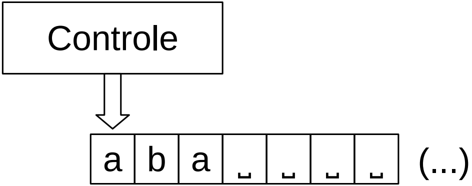
O filme - O Jogo da Imitação



Máquina de Turing

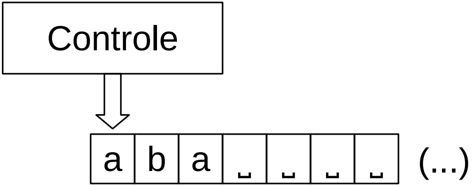
- A máquina de Turing é semelhante a um autômato finito, mas com memória ilimitada.
- É um modelo mais próximo de um computador de uso geral.
- A máquina de Turing faz o que um computador real é capaz de fazer.
- Ainda assim, a máquina de Turing não é capaz de resolver alguns problemas de computação (computabilidade).

Máquina de Turing



- O modelo da máquina de Turing usa uma fita ilimitada como sua memória.
- Ela tem um cabeçote que pode ler ou escrever símbolos na fita e mover-se sobre a fita.
- Inicialmente a fita contém a cadeia de entrada e o restante está em branco.
- Para armazenar informação a máquina escreve na fita.

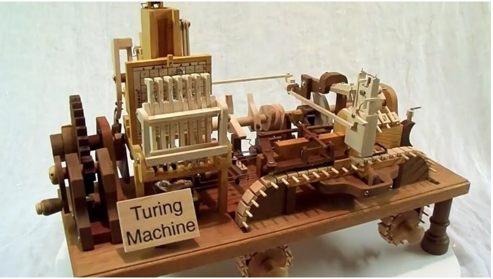
Máquina de Turing



- Para ler novamente a informação escrita a máquina pode mover a cabeça para a posição da fita onde a informação foi escrita.
- A máquina computa até que decide produzir uma saída. As saídas de *aceite* ou *rejeite* são obtidas quando a máquina muda para estados de aceitação e rejeição respectivamente.
- Se a máquina não entrar no estado de aceitação ou rejeição, continuará computando para sempre, sem parar.

Diferenças AF x MT

- MT pode ler e escrever sobre a fita.
- A cabeça de leitura pode mover-se tanto para direita como para esquerda.
- A fita é infinita.
- Os estados finais de aceitação e rejeição tem efeito imediatamente.



Um exemplo

- Para verificar informalmente o funcionamento da MT vamos construir uma máquina para verificar a pertinência de cadeias da linguagem:

$$B = \{w\#w | w \in \{0,1\}^*\}$$

- Suponha que a cadeia de entrada a ser verifica encontra-se já preenchida na fita. O objetivo é verificar se o que está escrito na fita é ou não membro da linguagem B, ou seja, se a entrada é composta de duas sequências idênticas de zeros e uns separados pelo símbolo $\#$.

Um exemplo

$M_1 =$ "Sobre a cadeia de entrada w:

1. Faça um zigue-zague ao longo da fita checando os símbolos correspondentes em ambos os lados do símbolo # para verificar as correspondências. Se os símbolos não são correspondentes então a MT deve rejeitar a entrada. Marque com um x os símbolos que forem verificados para manter o registro de quais são correspondentes.
2. Quando todos os símbolos à esquerda da # for marcado, verifique a existência de algum símbolo à direita de # que não foi marcado. Se existir então rejeite, senão aceite a entrada.

0	1	1	0	0	0	#	0	1	1	0	0	0	⊔	...

Um exemplo

M_1 = "Sobre a cadeia de entrada w:

0	1	1	0	0	0	#	0	1	1	0	0	0	$\sqcup$	...
x	1	1	0	0	0	#	0	1	1	0	0	0	$\sqcup$	...
x	1	1	0	0	0	#	x	1	1	0	0	0	$\sqcup$	...
x	1	1	0	0	0	#	x	1	1	0	0	0	$\sqcup$	...
x	x	1	0	0	0	#	x	1	1	0	0	0	$\sqcup$	...
x	x	x	x	x	x	#	x	x	x	x	x	x	$\sqcup$	...

Definição formal de uma MT

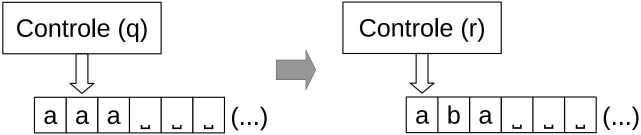
Uma **máquina de Turing** M é uma 7-upla, $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{aceita}, q_{rejeita})$, onde Q, Σ e Γ são conjuntos finitos e:

1. Q é o conjunto de **estados**
2. Σ é o **alfabeto de entrada** sem o símbolo branco $\sqcup$
3. Γ é o **alfabeto da fita**, onde $\sqcup \in \Gamma$ e $\Sigma \subseteq \Gamma$
4. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{E, D\}$ é a **função de transição**
5. $q_0 \in Q$ é o **estado inicial**
6. $q_{aceita} \in Q$ é o **estado de aceitação**
7. $q_{rejeita} \in Q$ é o **estado de rejeição**

A função de transição da MT

$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{E, D\}$

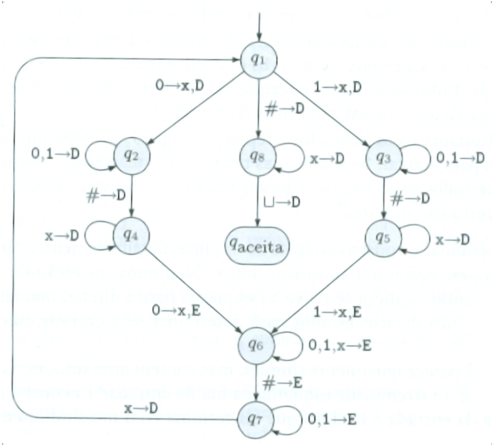
Se M está no estado q e a cabeça de leitura está sobre uma célula da fita contendo a, a transição $\delta(q, a) = (r, b, E)$ diz que M deve ir para o estado r, escrever b na fita e mover o cabeçote para **esquerda (E)**.



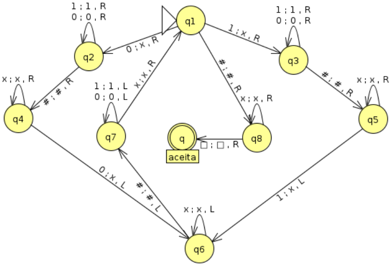
MT exemplo

MT para reconhecer $B = \{w\#w | w \in \{0,1\}^*\}$

1. $Q = \{q_1, q_2, ..., q_8, q_{aceita}, q_{rejeita}\}$.
2. $\Sigma = \{0, 1, \#\}$ e $\Gamma = \{0, 1, \#, x, \sqcup\}$
3. δ está representado no diagrama:



Função de transição



δ	0	1	$\#$	x	$\sqcup$
q_1	(q_2, x, D)	(q_3, x, D)	$(q_8, \#, D)$		
q_2	$(q_2, 0, D)$	$(q_2, 1, D)$	$(q_4, \#, D)$		
q_3	$(q_3, 0, D)$	$(q_3, 1, D)$	$(q_5, \#, D)$		
q_4	(q_6, x, E)			(q_4, x, D)	
q_5		(q_6, x, E)		(q_5, x, D)	
q_6			$(q_7, \#, E)$	(q_6, x, E)	
q_7	$(q_7, 0, E)$	$(q_7, 1, E)$		(q_1, x, D)	
q_8				(q_8, x, D)	$(q_{aceita}, \sqcup, D)$

Computação da MT

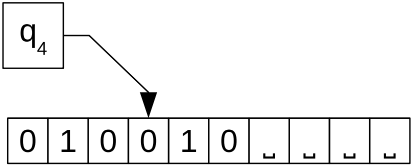
- A MT recebe a entrada $w = w_1 w_2 ... w_n \in \Sigma^*$ sobre as n primeiras células mais à esquerda da fita.
- A cabeça de leitura começa sobre w_1 e depois dos n símbolos estão símbolos em branco que não pertencem a Σ e por isso marcam o fim da cadeia de entrada.
- Se a cabeça tentar mover para uma posição mais à esquerda da primeira célula da fita, ela permanece na mesma posição.
- M continua até que encontra o estado de aceitação ou rejeição e se isso não acontece M continua para sempre.

Configuração

- Pode-se representar cada passo da computação da MT através de uma configuração instantânea.
- Para um estado q e duas cadeias $u, v \in \Gamma^*$, escrevemos uqv para configuração em que o estado atual é q, o conteúdo atual da fita é uv e a cabeça de leitura está sobre o primeiro símbolo de v.

Exemplo:

$$010q_4010$$



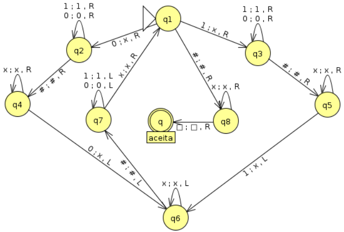
Configuração

Sejam $a, b \in \Gamma$, $u, v \in \Gamma^*$ e $q_i, q_j \in Q$

uaq_ibv	para $\delta(q_i, b) = (q_j, c, E)$ origina	uq_jacv
uaq_ibv	para $\delta(q_i, b) = (q_j, c, D)$ origina	$uacq_jv$

- Casos especiais ocorrem na extremidade da fita (início ou primeira posição de branco). q_ibv origina q_icv para um movimento para esquerda. uaq_i é equivalente a $uaq_i\sqcup$.
- A configuração inicial da MT é q_0w.
- A configuração de aceitação o estado é q_{aceita}.
- A configuração de rejeição o estado é $q_{rejeita}$.
- MT **aceita** a entrada w se temos a sequência de configurações $C_1, C_2, ...C_k$, onde:
  1. C_1 é a configuração inicial
  2. cada C_i origina C_{i+1} para $1 \leq i < k$
  3. C_k é a configuração de aceitação.

Exemplo de computação



$q_1 01\#01$

$xq_2 1\#01$

$x1q_2\#01$

$x1\#q_4 01$

$x1q_6\#x1$

$xq_7 1\#x1$

$q_7 x1\#x1$

$xq_1 1\#x1$

$xxq_3\#x1$

$xx\#q_5 x1$

$xx\#xq_5 1$

$xx\#q_6 xx$

$xxq_6\#xx$

$xq_7 x\#xx$

$xxq_1\#xx$

$xx\#q_8 xx$

$xx\#xq_8 x$

$xx\#xxq_8\sqcup$

$xx\#xx\sqcup\sqcup q_{aceita}$

Definições

Definição

Chame uma linguagem de **Turing-reconhecível** *(ou linguagem recursivamente enumerável), se alguma máquina de Turing a reconhece.*

A MT reconhece a linguagem aceitando ou rejeitando a cadeia de entrada, mas eventualmente pode entrar em loop.

Definição

Chame uma linguagem de **Turing-decidível** *(ou linguagem recursiva), se alguma máquina de Turing a decide.*

A MT sempre para com uma decisão de aceitar ou rejeitar a cadeia para a linguagem.

Exemplo 1

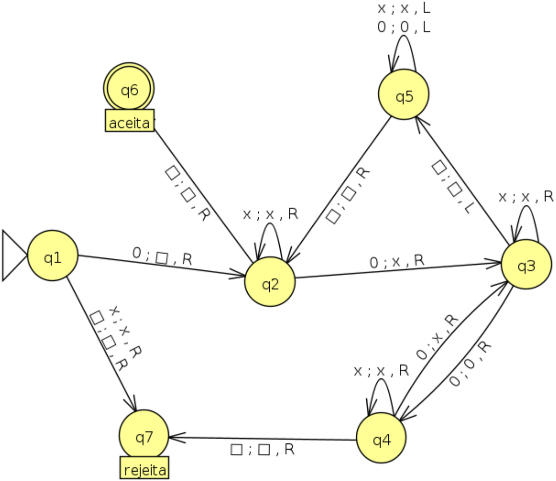
M_1 que decide $A = \{0^{2^n} | n \geq 0\}$

M_1 = "Sobre a cadeia de entrada w:

1. Faça uma varredura da esquerda para direita na fita, marcando um 0 sim e outro não.
2. Se no passo 1, a fita continha um único 0, *aceite*.
3. Se no passo 1, a fita continha mais que um único 0 e o número de 0s era ímpar, *rejeite*.
4. Retorne a cabeça de leitura para a extremidade esquerda da fita.
5. Vá para o passo 1."

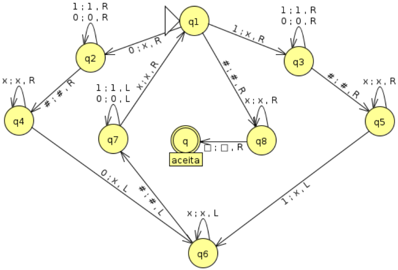
Exemplo 1

M_1 que decide $A = \{0^{2^n} | n \geq 0\}$



Exemplo 2

M_2 que decide $B = \{w\#w | w \in \{0,1\}^*\}$



Exemplo 3

M_3 que decide $C = \{a^i b^j c^k | i \times j = k \text{ e } i, j, k \geq 1\}$

M_3 = "Sobre a cadeia de entrada w:

1. Faça uma varredura da esquerda para direita na fita, para verificar se ela é membro de $a^+ b^+ c^+$, *rejeite* se não for.
2. Retorne a cabeça para a extremidade esquerda da fita.
3. Marque um a e faça uma varredura para a direita até encontrar um b. Vá e volte entre b's e c's, marcando um de cada até que todos os b's tenham terminado. Se todos os c's forem marcados e tiverem b's não marcados, *rejeite*.
4. Restaure os b's marcados e repita o passo 3, se existe algum outro a para marcar. Se todos os a's foram marcados, verifique se todos os c's também foram marcados. Se sim, *aceite*, senão *rejeite*."

Agenda

Variantes de MT

- Existem definições alternativas para Máquinas de Turing como MT com múltiplas fitas ou MT com indeterminismo.
- A MT e suas variantes tem o mesmo poder, ou seja, reconhecem as mesmas linguagens.
- Chamamos de **Robustez** a invariância da máquina de Turing a certas mudanças.
- Suponha uma MT que além de mover a cabeça para esquerda e direita, também pudesse ficar parada. A função de transição teria a forma: $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{E, D, P\}$
- Podemos implementar essa opção (ficar parada) numa MT qualquer substituindo essa transição por duas, uma que move para direita e outra que move para esquerda.

MT Multifita

- Uma MT multifita é como uma MT comum, com a variação de possuir múltiplas fitas. Cada fita tem sua própria cabeça de leitura e escrita. Inicialmente a entrada aparece sobre a fita 1 e todas as outras iniciam em branco.
- A função de transição é modificada para permitir a leitura, escrita e movimentação nas fitas simultaneamente:

$$\delta : Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{E, D, P\}^k$$

- A transição:

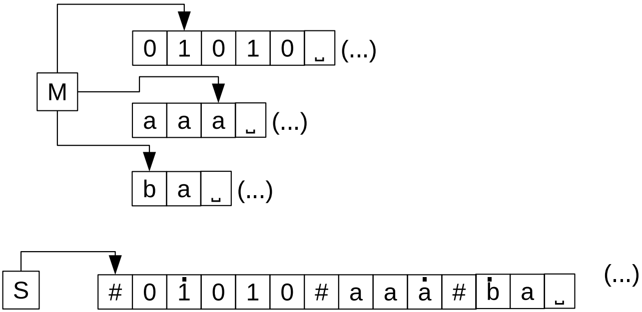
$$\delta(q_i, a_1, ...a_k) = (q_j, b_1, ..., b_k, E, D, ..., E)$$

Significa que se a MT está no estado q_i e lê nas fitas os símbolos $a_1, ..., a_k$, ela muda para o estado q_j, escreve nas fitas os símbolos $b_1, .., b_k$ e vai para direita, esquerda ou fica parada, conforme especificado.

MT Multifita

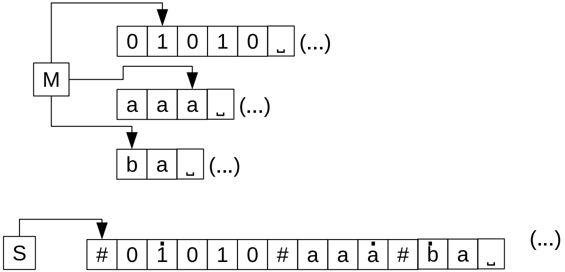
Teorema

Toda máquina de Turing Multifita (M) tem uma máquina de Turing de uma única fita (S) que lhe é equivalente.



Simulação em S

- S simula o efeito de k fitas armazenando sua informação na sua única fita.
- Usa-se o # como delimitador para separar o conteúdo de cada fita.
- Usa-se símbolos especiais (com um ponto acima) para representar a posição da cabeça em cada fita.



MT Não-Determinística

- Uma MT Não-Determinística é como uma MT comum, com a variação de que a qualquer ponto em uma computação, a máquina pode proceder de acordo com várias possibilidades.
- A função de transição é modificada da seguinte forma:

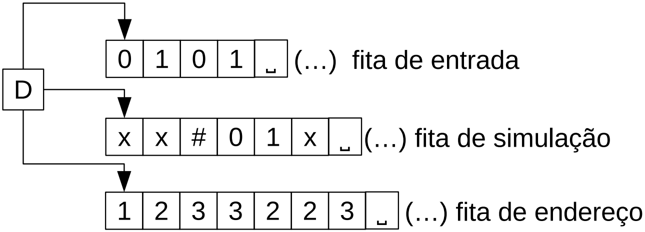
$$\delta : Q \times \Gamma \rightarrow P(Q \times \Gamma \times \{E, D, P\})$$

- A computação de uma máquina de Turing não-determinística é uma árvore em que cada ramo corresponde a uma configuração de computação diferente da MT.

MT Não-Determinística

Teorema

Toda máquina de Turing Não-Determinística (N) tem uma máquina de Turing determinística (D) que lhe é equivalente.



Enumeradores

- Usamos a terminologia *linguagem recursivamente enumerável* para linguagem Turing-reconhecível.
- Esse termo origina de uma máquina equivalente a máquina de turing chamado **enumerador**.
- Vagamente, um enumerador é uma MT com uma impressora, para imprimir as cadeias que ele reconhece.

Teorema

Uma linguagem é Turing-reconhecível se e somente se algum enumerador a enumera.

Equivalência com outros modelos

- Esses diversos modelos são equivalentes em poder, desde que eles satisfaçam a requisitos razoáveis.
- Uma analogia com linguagens de programação: Haskell e C.
  - Ambas tem estrutura e estilos diferentes.
  - Todo algoritmo pode ser implementado numa ou noutra linguagem.
  - Ainda que alguns algoritmos sejam mais fáceis de expressar em Haskell ou em C.

Agenda

Algoritmos

- É uma coleção de operações simples para realizar um tarefa.
- Muito importante na matemática (calculistas!)
- A definição precisa dos algoritmos foi crucial para um importante problema da matemática.
    - O décimo problema de Hilbert (1900) - conceber um algoritmo (*um processo com o qual ela possa ser determinada por um número finito de operações*) que testasse se um polinômio tinha um raiz inteira.

$$6x^3yz + 2xy^2 - x^3 - 10$$

    - A definição para algoritmos veio no sistema de λ-cálculos de Church e nas "máquinas"de Turing (1936).
    - A conexão entre a noção informal de algoritmos e a definição precisa é chamada **"tese de Church-Turing"**.

Terminologias para MTs

- Podemos padronizar a forma de descrever algoritmos para máquinas de Turing:
  - **Descrição formal** que esmiúça todos os detalhes da MT (estados, alfabetos, função de transição)
  - **Descrição da implementação** que usa a linguagem natural para descrever como a máquina de Turing move a cabeça e executa leituras e escritas para resolver o problema.
  - **Descrição de alto-nível** que usa a linguagem natural para descrever o algoritmo, sem se preocupar com os detalhes da implementação.

Exemplo

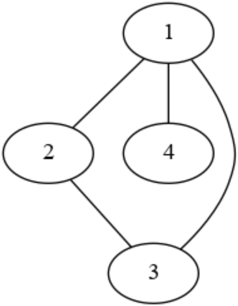
Seja A a linguagem consistindo de todas as cadeias representando grafos não-direcionados que são conexos.

$$A = \{< G > | G \text{ é um grafo não direcionado conexo } \}$$

$M_3 =$ "Sobre a entrada $< G >$, a representação de um grafo G:

1. Selecione o primeiro nó de G e marque-o.
2. Repita o seguinte passo até que nenhuma novo nó seja marcado
    2.1 Para cada nó de G, marque-o, se ele estiver ligado por uma aresta a um nó que já está marcado.
3. Faça uma varredura em todos os nós de G para determinar se eles estão todos marcados. Se sim, *aceite*, senão *rejeite*."

Grafo e sua representação



$$< G >= (1, 2, 3, 4)((1, 2), (2, 3), (3, 1), (1, 4))$$

Agenda

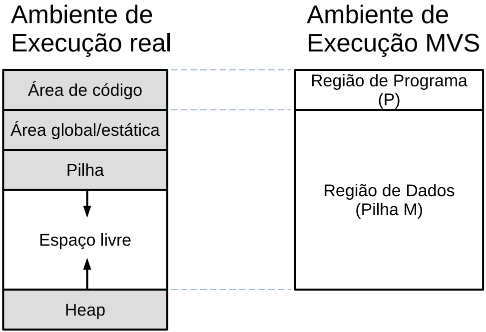
Máquina Virtual

- O objetivo principal do compilador para uma linguagem de programação é construir uma ferramenta de tradução que transforme os códigos numa linguagem de alto nível para instruções numa linguagem de máquina para, assim, possibilitar a sua execução.

- A arquitetura de uma máquina real é complexa, o que torna essa tarefa trabalhosa.
- Para simplificar, definiremos uma máquina virtual mais simples (MVS), mais conveniente para um compilador didático.

Características da MVS

- A Máquina Virtual Simples (MVS) é uma máquina baseada na Máquina de Execução Pascal (MEPA), proposta por Tomás Kowaltowski.
- A memória da MVS é composta de duas regiões:
  1. A região de programa P que conterá as instruções do programa em MVS que a máquina está executando.
  2. A região de pilha de dados M que conterá os valores manipulados pelas instruções MVS.



Características da MVS

As regiões de memória P e M funcionam como vetores com índices numerados de zero até uma tamanho máximo. Além disso, MVS tem três registradores. São eles:

1. O registrador i contém o endereço da próxima instrução a ser executada, $P[i]$. Esse registrador é incrementado automaticamente após a execução de cada instrução. Exceto para as instruções de desvio, que alteram de forma absoluta o valor de i.
2. O registrador s indicará o elemento no topo da pilha de dados, $M[s]$.
3. O registrador de base d que contém o endereço de base no qual a variável está inserida. Esse único registrador é suficiente para generalizar a forma de endereçamento das variáveis locais e globais no programa. Usaremos endereçamento indireto para variáveis locais e globais.

Instruções MVS

#	Instrução	Micro-código
1	CRCT k	$s \leftarrow s + 1$ $M[s] \leftarrow k$
2	CRVG n	$s \leftarrow s + 1$ $M[s] \leftarrow M[n]$
3	ARZG n	$M[n] \leftarrow M[s]$ $s \leftarrow s - 1$
4	SOMA	$M[s-1] \leftarrow M[s-1] + M[s]$ $s \leftarrow s - 1$
5	SUBT	$M[s-1] \leftarrow M[s-1] - M[s]$ $s \leftarrow s - 1$
6	MULT	$M[s-1] \leftarrow M[s-1] * M[s]$ $s \leftarrow s - 1$
7	DIVI	$M[s-1] \leftarrow M[s-1]/M[s]$ $s \leftarrow s - 1$
8	CMMA	$\underline{\text{SE}}\ M[s-1] > M[s]\ \underline{\text{ENTAO}}\ M[s-1] \leftarrow 1\ \underline{\text{SENAO}}\ M[s-1] \leftarrow 0;$ $s \leftarrow s - 1$
9	CMME	$\underline{\text{SE}}\ M[s-1] < M[s]\ \underline{\text{ENTAO}}\ M[s-1] \leftarrow 1\ \underline{\text{SENAO}}\ M[s-1] \leftarrow 0$ $s \leftarrow s - 1$
10	CMIG	$\underline{\text{SE}}\ M[s-1] = M[s]\ \underline{\text{ENTAO}}\ M[s-1] \leftarrow 1\ \underline{\text{SENAO}}\ M[s-1] \leftarrow 0$ $s \leftarrow s - 1$

Instruções MVS

#	Instrução	Micro-código
11	DISJ	SE $M[s-1]$ ou $M[s]$ ENTAO $M[s-1] \leftarrow 1$ SENAO $M[s-1] \leftarrow 0$ $s \leftarrow s-1$
12	CONJ	SE $M[s-1]$ e $M[s]$ ENTAO $M[s-1] \leftarrow 1$ SENAO $M[s-1] \leftarrow 0$ $s \leftarrow s-1$
13	NEGA	$M[s] \leftarrow 1 - M[s]$
14	DSVS p	$i \leftarrow p$
15	DSVF p	SE $M[s] = 0$ ENTAO $i \leftarrow p$ SENAO $i \leftarrow i+1$ $s \leftarrow s-1$
16	LEIA	$s \leftarrow s+1$ "$M[s] \leftarrow$ Entrada"
17	ESCR	"Escreve $M[s]$ " $s \leftarrow s-1$
18	NADA	"Não faz nada"
19	INPP	$s \leftarrow -1$ $i \leftarrow 1$ $D \leftarrow 0;$
20	FIMP	"Finaliza a execução"
21	AMEM n	$s \leftarrow s+n$

Tradução de Expressões

- Como a máquina MVS é baseada em pilha, as expressões em notação infixa (com o operação entre os operandos) da linguagem fonte devem ser traduzidas para uma sequência de instruções em NPR (Notação Polonesa Reversa, na qual a operação é colocada após os seus operandos).
- Considere a expressão B * (A + 30) - A e suponha que os endereços atribuídos pelo compilador as variáveis A e B são 1 e 3, respectivamente. Considere ainda que o valor da variável A é 10 e que o valor da variável B é 20. A tradução é:

```
CRVG  3
CRVG  1
CRCT  30
SOMA
MULT
CRVG  1
SUBT
```

Execução da Expressão

Expressão: B * (A + 30) - A, onde A = 10 e B = 20

8	?	?	?	?	?	?	?	?
7	?	?	?	**30** ← s	30	30	30	30
6	?	?	**10** ← s	10	**40** ← s	40	**10** ← s	10
5	?	**20** ← s	20	20	20	**800** ← s	800	**790** ← s
4	? ← s	?	?	?	?	?	?	?
3	20	20	20	20	20	20	20	20
2	?	?	?	?	?	?	?	?
1	10	10	10	10	10	10	10	10
0	?	?	?	?	?	?	?	?
		CRVG 3	CRVG 1	CRCT 30	SOMA	MULT	CRVG 1	SUBT

Tradução da Atribuição

Instrução	Micro-código
ARZG n	$M[n] \leftarrow M[s]$ $s \leftarrow s - 1$

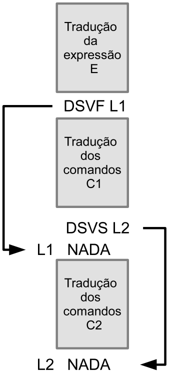
- Considere o comando de atribuição $\underline{A \leftarrow A + B}$, onde os endereços das variáveis A e B são 10 e 12 respectivamente. A tradução MVS para essa atribuição é:

```
1   CRVG  10
2   CRVG  12
3   SOMA
4   ARZG  10
```

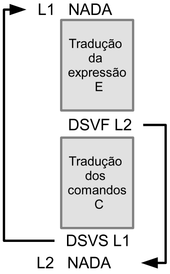
Tradução da Repetição e Seleção

Instrução	**Micro-código**
DSVS p	$i \leftarrow p$
DSVF p	<u>SE</u> $M[s] = 0$ <u>ENTAO</u> $i \leftarrow p$ <u>SENAO</u> $i \leftarrow i + 1$ $s \leftarrow s - 1$
NADA	"Não faz nada"

se E então C1 senao C2 fimse

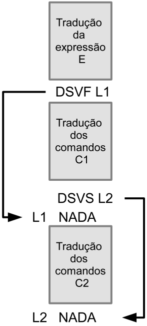


enquanto E faca C fimenquanto



Tradução da Seleção

```
A <- V
se A
  entao A <- F
  senao A <- V
fimse
```

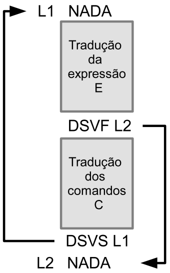


A tradução MVS é:

```
       CRCT  1
       ARZG  0
       CRVG  0
       DSVF  L1
       CRCT  0
       ARZG  0
       DSVS  L2
  L1   NADA
       CRCT  1
       ARZG  0
  L2   NADA
```

Tradução da Repetição

```
x <- 1
enquanto x < 10 faca
   x <- 2 * x
fimenquanto
```



A tradução MVS é:

```
        CRCT  1
        ARZG  0
    L1  NADA
        CRVG  0
        CRCT  10
        CMME
        DSVF  L2
        CRCT  2
        CRVG  0
        MULT
        ARZG  0
        DSVS  L1
    L2  NADA
```

Tradução de Leitura e Escrita

Instrução	Micro-código
LEIA	$s \leftarrow s + 1$ "$M[s] \leftarrow$ Entrada"
ESCR	"Escreve $M[s]$ " $s \leftarrow s - 1$

```
1       leia  A
2       leia  B
3       escreva  A + B
```

A tradução MVS é:

```
1       LEIA
2       ARZG  0
3       LEIA
4       ARZG  1
5       CRVG  0
6       CRVG  1
7       SOMA
8       ESCR
```

Tradução de Programa

Para tradução de um programa em linguagem Simples completo usaremos as seguintes instruções MVS:

Instrução	**Micro-código**
INPP	$s \leftarrow -1$ $i \leftarrow 1$ $d \leftarrow 0$
FIMP	"Finaliza a execução"
AMEM n	$s \leftarrow s + n$

Onde:

- INPP - instrução MVS que serve para colocar a máquina de execução numa configuração inicial;
- FIMP - instrução que finaliza a execução de um programa.
- AMEM - instrução que aloca memória para as variáveis globais do programa.

Exemplo de tradução

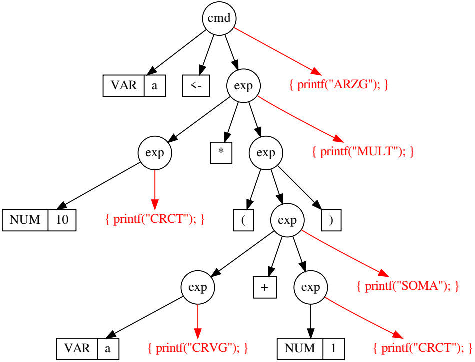
```
programa repete
  inteiro i j
inicio
  i <- 1
  enquanto i < 10 faca
    j <- 1
    enquanto j < 10 faca
      escreva i + j
      j <- j + 1
    fimenquanto
    i <- i + 1
  fimenquanto
fimprograma
```

```
    INPP
    AMEM 2
    CRCT 1
    ARZG 0
L1  NADA
    CRVG 0
    CRCT 10
    CMME
    DSVF L2
    CRCT 1
    ARZG 1
L3  NADA
    CRVG 1
    CRCT 10
    CMME
    DSVF L4
    CRVG 0
    CRVG 1
    SOMA
    ESCR
    CRVG 1
    CRCT 1
    SOMA
    ARZG 1
    DSVS L3
L4  NADA
    CRVG 0
    CRCT 1
    SOMA
    ARZG 0
    DSVS L1
L2  NADA
    FIMP
```

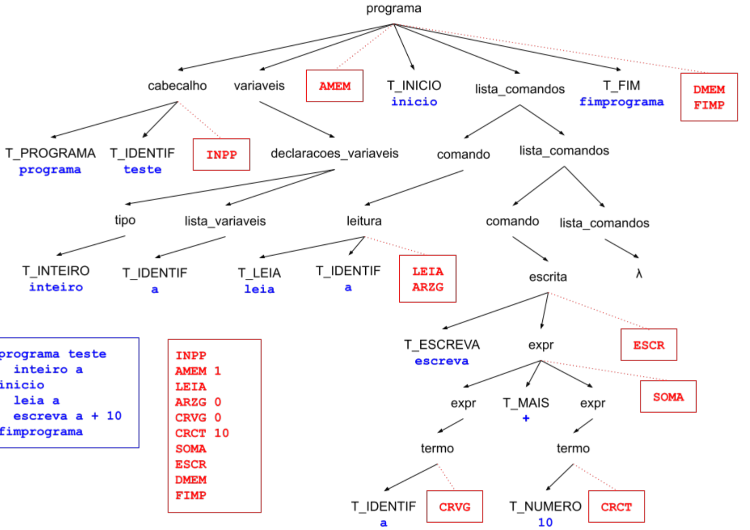
Geração de código MVS (calculadora)

```
cmd  :
     expr                 { }
   | VAR ATRIBUI expr
        {printf("\tARZG\t%d\n",$1);}
   ;

expr  :
     NUM
        {printf("\tCRCT\t%d\n",$1);}
   | VAR
        {printf("\tCRVG\t%d\n",$1);}
   | expr MAIS expr
        {printf("\tSOMA\n");}
   | expr MENOS expr
        {printf("\tSUBT\n");}
   | expr BARRA expr
        {printf("\tDIVI\n");}
   | expr VEZES expr
        {printf("\tMULT\n");}
   | ABRE expr FECHA    { }
   ;
```



Geração de código




```
programa teste
  inteiro a
inicio
  leia a
  escreva a + 10
fimprograma
```

```
INPP
AMEM 1
LEIA
ARZG 0
CRVG 0
CRCT 10
SOMA
ESCR
DMEM
FIMP
```