Máquina de Turing Binária

Henrique S. Souza, Mateus Kosicov

1 de junho de 2025

1 Introdução

Esse relatório documenta a implementação de uma Máquina de Turing Binária (MTB), em assembly ARM. Formalmente, define-se aqui a MTB pela uma 7-upla $M = (Q, \Gamma, \Sigma, \delta, \square, q_0, q_{\text{halt}})$, na qual:

Q: Conjunto finito de estados; $Q = \{q_0, q_1, q_2, ..., q_h\}$.

Γ: Alfabeto da fita: $\Gamma = \{0, 1, \square\}$.

$\Sigma \subset \Gamma \backslash \{\square\}$: Alfabeto de entrada; $\Sigma = \{0, 1\}$.

$\square$: símbolo de espaço em branco (blank symbol).

δ: Função de transição:$\delta : (Q \backslash \{q_{halt}\}) \times \Gamma \to Q \times \Gamma \times \{\leftarrow, \rightarrow\}$.

$q_0 \in Q$: Estado inicial.

$q_{\text{h}} \in Q$: Estado de HALT.

Na implementação mais genérica, o Γ pode conter um variedade maior de símbolos e há dois estados de HALT q_{acc} e q_{rej}. Na BTM proposta aqui, propôs-se um alfabeto Γ minimalista, somente com dígitos binários (0 e 1) e o blank symbol ($\square$), e condensou os estados de aceitação e rejeição para um mesmo estado HALT, denominado q_h.

Representa-se cada símbolo $\gamma \in \Gamma$ do alfabeto Γ por dois bits $\gamma = \gamma_0\gamma_1$, segundo a convenção:

Bits de $\gamma = \gamma_0\gamma_1$	**Símbolo**
00	0
01	1
11	$\square$

Tabela 1: Codificação Binária dos Símbolos da Fita (Γ)

Os movimentos do cabeçote $\{\leftarrow, \rightarrow\}$ são representados por um único bit d em que $\leftarrow$ corresponde a $d = 0$ e $\rightarrow$ corresponde a $d = 1$.

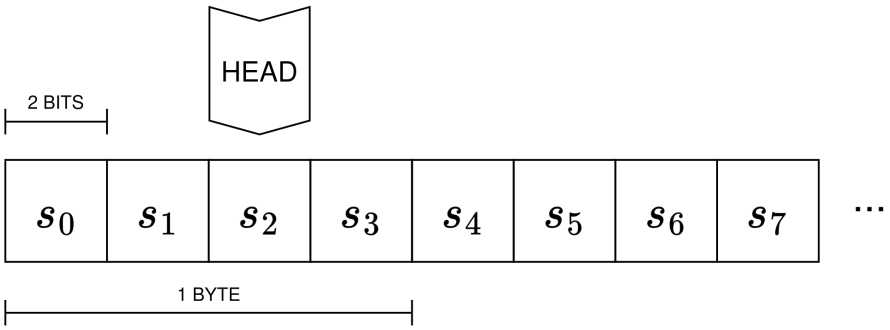

Figura 1: Esquema lógico da BTM

2 Organização de Memória

Considera-se quatro regiões de memória para o programa BTM:

PROGRAM: o código binário de controle da BTM

HEADER: contém informações sobre a localização das regiões de memória da máquina de estados finita (FSM) e da fita binária, além de alguns dados essenciais como tamanho da fita e posição inicial do cabeçote.

FSM: contém os estados $q \in Q$ da FSM e descreve a função de transição δ.

TAPE: contém a fita binária da BTM.

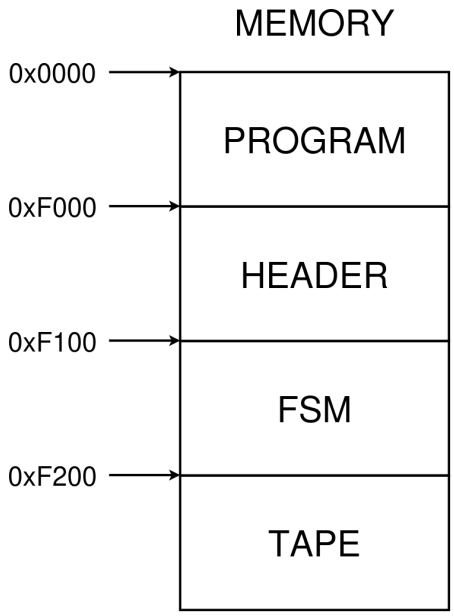

Figura 2: Organização de Memória da BTM

Por convenção, as regiões HEADER, FSM TAPE começam nos endereços 0xF000, 0xF100 e 0xF200. Esses valores podem ser alterados, por exemplo, para minimizar o desperdício de espaço entre uma região e outra.

2.1 Header

O HEADER é a região de memória responsável por guardar informações básicas da BTM. Por padrão, ele começa no endereço 0x0000F000 e contém as informações:

fsm_addr (4 bytes): ponteiro para o início da FSM

tape_addr (4 bytes): ponteiro para o início da fita

tape_size (4 bytes): tamanho da fita

start_pos (4 bytes): posição do cabeçote na fita, representada por índice inteiro.

2.2 FSM

Quando o cabeçote lê o símbolo $x \in \Gamma$, a depender do estado atual da FSM $q \in Q$, ele deve substituí-lo pelo símbolo $y \in \Gamma$ e mover-se na direção $d \in \{\leftarrow, \rightarrow\}$, e transitar para o estado q^*, segundo a função de transição de estados. Para capturar esse comportamento, que engloba tanto o conjunto de estados Q quanto a função de transição δ, propõe-se uma representação compacta da lógica de execução da BTM para o estado q_i como a 6-upla $y_0, y_1, d_0, d_1, q_0^*, q_1^*$, em que:

y_0 (1 bit): dígito a ser escrito na fita, caso $x = 0$

y_1 (1 bit): dígito a ser escrito na fita, caso $x = 1$

d_0 (1 bit): próximo movimento do cabeçote, caso $x = 0$

d_1 (1 bit): próximo movimento do cabeçote, caso $x = 1$

q_0^* (1 byte): índice do próximo estado da FSM, caso $x = 0$

q_1^* (1 byte): índice do próximo estado da FSM, caso $x = 1$

Nesse modelagem, cada bloco $y_0, y_1, d_0, d_1, q_0^*, q_1^*$ determina um estado q_i, sua transição e o movimento do cabeçote; o conjunto desses blocos forma a FSM completa. Na memória, (y_0, y_1, d_0, d_1) são representados por um único byte no formato $(0000 y_0 y_1 d_0 d_1)$, em que os 4 bits mais significativos são zerados e os 4 bits menos significativos são os parâmetros da lógica do estado. Como os índices dos próximos estados da FSM ocupam cada 1 byte, no total, cada bloco $y_0, y_1, d_0, d_1, q_0^*, q_1^*$ ocupa 3 bytes de memória.

Há uma exceção para o caso de halting, em que o cabeçote deve parar, mas, a princípio, esse problema será resolvido na lógica de controle do cabeçote, não na modelagem da FSM em si.

2.3 Fita

A fita está delimitada em uma região de memória especificada pelo HEADER. Ela é uma sequência de símbolos $s_0, s_1, s_2, \cdots, s_n$, com cada $s_i \in \Gamma$ representada pelos bits $\gamma_0\gamma_1$. Por exemplo, a fita $F = (\square, \square, 0, 1, 1, 0, 1, \square)$ é representada como 0xF147 = 1111000101000111.

3 Exemplo

Considera-se como exemplo uma BTM programada para detectar a sequência 11 na fita binária. Ela movimenta-se sempre para a direita, inverte os símbolos 0 e 1 e, ao passar pela sequência 11, entra no estado de halt q_h.

3.1 FSM

Para esse programa, propôs-se a FSM descrita pela figura 3, com os estados $Q = \{q_0, q_1, q_h\}$.

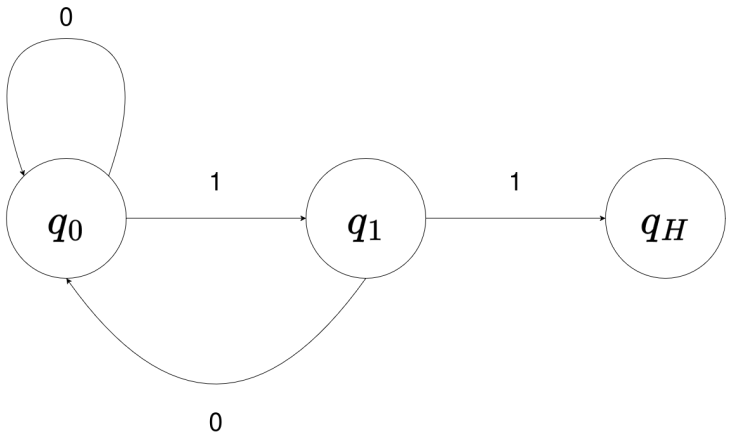

Figura 3: FSM da BTM para detecção de sequência 11

Assim, de acordo com a modelagem proposta, os estados e suas transições estão descritos pelas tabelas 3.1 e 3.1:

Estado	índice (dec)	índice (hex)
q_0	0	0x00
q_1	1	0x01
q_h	255	0xFF

Tabela 2: Codificação dos estados da FSM exemplo

Estado	y_0	y_1	d_0	d_1	q_0^*	q_1^*
q_0	1	0	1	1	0x00	0x01
q_1	1	0	1	1	0x00	0xFF

Tabela 3: Codificação Binária dos Símbolos da Fita (Γ)

Quanto aos estados, note que o índice de q_h é sempre 0xFFFF; para q_0 e q_1, os índices adotados são, naturalmente, 1 e 2. De acordo com a tabela 3.1, $y_0 = 1$ e $y_1 = 0$ para q_0 e q_1 significa que, nos dois estados, o cabeçote escreve $y = 0$ se $x = 1$ e $y = 1$ se $x = 0$ (inverte-se

1 e 0 na fita). Além disso, $d_0 = d_1 = 1$ implica que o cabeçote sempre movimenta-se para a direita, qualquer que seja seu estado.

Na memória em si, armazenamos os estados concatenando os 6 bytes abaixo:

Byte 0 de q_0: 00001011 = 0x0B (contém $y_0 y_1 d_0 d_1$)

Byte 1 de q_0: 00000000 = 0x00 (próximo estado caso $x = 0$)

Byte 2 de q_0: 00000001 = 0x01 (próximo estado caso $x = 1$)

Byte 0 de q_1: 00001011 = 0x0B (contém $y_0 y_1 d_0 d_1$)

Byte 1 de q_1: 00000000 = 0x00 (próximo estado caso $x = 0$)

Byte 2 de q_1: 11111111 = 0xFF (próximo estado caso $x = 1$)

Portanto, armazena-se 0x0B0001 0B00FF (3 bytes) na memória. No entanto, repare que, por conta da endianness do ARM, devemos armazenar, na realidade, 0x0B01000B 0000FF00.

3.2 Fita

Para fita de exemplo, usaremos

$$F = (\square, \square, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0)$$

Cujo cabeçote inicia na posição $i = 0$ (primeiro símbolo da fita). Codificando os símbolos de acordo com a convenção proposta aqui, temos:

i	s_i	$\gamma_0\gamma_1$	i	s_i	$\gamma_0\gamma_1$
0	$\square$	11	8	0	00
1	$\square$	11	9	1	01
2	0	00	10	1	01
3	0	00	11	0	00
4	1	01	12	0	00
5	0	00	13	1	01
6	0	00	14	0	00
7	1	01	15	0	00

Tabela 4: Codificação binária da fita exemplo

Reunindo as codificações $\gamma = \gamma_0\gamma_1$ de 4 em 4 símbolos, geramos 4 bytes:

Byte 0: 1111 0000 = 0xF0 (símbolos de $i = 0$ até $i = 3$)

Byte 1: 0100 0001 = 0x41 (símbolos de $i = 4$ até $i = 7$)

Byte 2: 0001 0100 = 0x14 (símbolos de $i = 8$ até $i = 11$)

Byte 3: 0001 0000 = 0x10 (símbolos de $i = 12$ até $i = 15$)

Concatenando-os, a fita F deve ser codificada como 0xF0411410; na endianness correta, isso se traduz na memória em 0x101441F0.

Repare que se a fita é lida da esquerda para a direita, a começar por $i = 0$, a BTM deve entrar em halt após passar por $i = 10$ na fita, em que está o segundo 1 do primeiro grupo 11. Então, até $i = 10$, os símbolos $x = 0$ e $x = 1$ devem estar trocados, mas, de $i = 11$ até $i = 15$, os símbolos devem ser preservados.

Ou seja, a fita tornará-se em F' dada por:

$$F' = (\square, \square, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0)$$

Codificada, na convenção proposta, como:

Byte 0: 1111 0101 = 0xF5 (símbolos de $i = 0$ até $i = 3$)

Byte 1: 0001 0100 = 0x14 (símbolos de $i = 4$ até $i = 7$)

Byte 2: 0100 0000 = 0x40 (símbolos de $i = 8$ até $i = 11$)

Byte 3: 0001 0000 = 0x10 (símbolos de $i = 12$ até $i = 15$)

O que resulta em F' codificada como 0xF5144010, que deve ser escrita na memória como 0x104014F5.

3.3 Teste

Efetuou-se o teste da BTM proposta no CPUlator, para o exemplo proposto, de detecção da sequência 11 e inversão dos símbolos $x = 0$ e $x = 1$ entre si, antes do estado de HALT q_0.

Para o exemplo, se a BTM estiver de fato correta, espera-se que, após a passagem pelo grupo 11 localizado em $i = 10$ na fita, a ela entre no estado HALT q_h e a fita deve ser substituída, de 0xF0411410 para 0xF5144010.

Inseriu-se no código subrotinas para escrita da FSM e da fita na memória. A FSM foi programada no endereço 0xF100, com os valores 0x0B01000B 0000FF00 (little endian) e a fita original foi escrita em 0xF200 com os valores 0x101441F0 (little endian), conforme as figuras 4 e 5, respectivamente.

Figura 4: FSM do exemplo programada em 0xF100

Figura 5: Fita programada em 0xF200

Em seguida, executou-se o código da BTM. A fita F' obtida após o processo foi 0x104014F5 (little endian), conforme demonstra afigura 6, que é exatamente o valor correto para a fita, previsto na secção anterior.

Figura 6: Fita após a execução da BTM no exemplo proposto

Além disso, comprovou-se que a máquina, de fato, entrou no estado de HALT q_h, pois o registrador r1 foi responsabilizado por armazenar o índice do estado atual e, conforme se observa na figura 7, seu valor final, após a execução do programa, foi exatamente 0xFF, que é o índice do estado de HALT definido no programa.

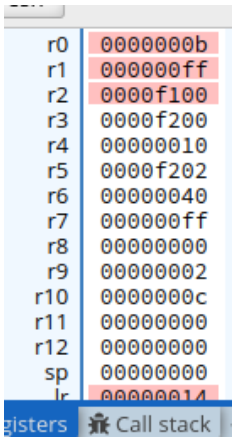

Figura 7: Registradores após a execução da BTM sobre o exemplo. Registrador r1 contém 0xFF, estado de HALT.

Dessa forma, verificou-se o adequado funcionamento da BTM para o exemplo proposto, logo, ela foi aprovada no teste.