
INTRODUÇÃO À CIÊNCIA DA COMPUTAÇÃO I
SCC0221

Cleyton, o Punk de 2077

1 Introdução

Em 2060, a humanidade previu que o Sol liberaria uma onda de radiação que mataria toda a humanidade e destruiria muito da tecnologia da Terra, essa onda seria inciada por volta de 2070, e nesses 10 anos os humanos prepararam um abrigo nuclear que isolaria toda a radiação durante a onda, para depois poderem voltar a habitar o planeta. Supla, o cientista principal envolvido nesse processo se preocupava com um detalhe: o sistema do abrigo era baseado em computadores externos, e por isso nenhum humano poderia reabrir os abrigos em 2077 — ano previsto para a volta de um grau não fatal de radiação.

Após pensar muito, Supla teve uma ideia promissora. De acordo com seus cálculos os únicos seres que sobreviveriam a tanta radiação seriam as baratas, sendo assim, o cientista escolheu um desses seres para ser seu aprendiz e o responsável pela abertura de portas em 2077. A barata escolhida foi Cleyton, por sua clara superdotação. Cleyton foi treinado por Supla nos laboratório do ICMC, em São Carlos, um dos lugares que sobreviveriam a radiação e por isso um lugar onde Cleyton poderia continuar seus estudos sozinhos enquanto Supla estivesse no abrigo.

Nesse tempo, enquanto ensinava Cleyton o básico sobre computação, Supla desenvolveu uma linguagem simplificada que auxiliaria a barata a abrir as portas em 2077, essa linguagem foi chamada de tape — nome inspirado pelo ritmo que musicava os estudos no laboratório, punk tocado em um toca fita. Chegado 2060, Supla ainda não havia terminado o manual para a nova linguagem, e deixou seu aprendiz responsável finalizar esse processo e desenvolver o aparelho que abriria os abrigos dali 7 anos.

2061 chegou, e agora restaram apenas o toca disco e Cleyton, o Pupilo Punk (apelido dado por seu professor, atualmente refugiado no abrigo), dado isso a barata parte seus estudos sobre a nova linguagem. Cleyton precisa desenvolver um interpretador, e para isso começa a investigar a parte do manual já escrita por Supla, descobrindo assim como a linguagem funciona.

A linguagem tape é baseada em programas descritos por uma fita contendo 512 números inteiros, esses valores representam operações, argumentos, ou dados. Tem-se um ponteiro de execução, o qual itera pela fita durante a execução do programa. No primeiro elemento da fita está a primeira operação do respectivo programa, esse elemento é seguido pelos argumentos dessa operação (caso ela requisite), após a execução dessa operação o ponteiro de execução estará apontando para operação seguinte. Esse ciclo se repete até que se encontre uma operação de parada (hlt - 0). As operações suportadas pela linguagem e seus respectivos funcionamentos foram descritas pelo cientista por meio de tabelas esquemáticas, veja a seguir:

1.1 Operação 0 - HALT (HALT)

Essa é a operação de parada do programa, representada pelo inteiro 0. Quando um 0 é lido como operação o interpretador deve encerrar a execução.

1.2 Operação 1 - ADD

Essa é a operação de adição entre dois inteiros escritos em endereços (índices começando em 0) da fita, representada pelo inteiro 1. Quando um 1 é lido como operação, o interpretador deve ler os próximos três elementos como argumentos. Esses argumentos representam, respectivamente, o endereço do primeiro inteiro da adição, o endereço do segundo inteiro da adição, e o endereço onde deve ser escrito o resultado da soma. Após esse processo, o ponteiro de execução se encontra no elemento seguinte ao último argumento. Veja a tabela contida no manual:

Operação 1: ADD											
	0	1	2	3	4	5	6	7	8	9	10
Antes	1	5	6	7	0	-3	5	4	0	0	-1
<code>add <add1> <add2> <dest></code>											
	0	1	2	3	4	5	6	7	8	9	10
Depois	1	5	6	7	0	-3	5	2	0	0	-1
Atribuir ao endereço <dest> o valor da soma dos conteúdos de <add1> e <add2>											

1.3 Operação 2 - MUL (MULTIPLY)

Essa é a operação de multiplicação entre dois inteiros escritos em endereços (índices começando em 0) da fita, representada pelo inteiro 2. Quando um 2 é lido como operação, o interpretador deve ler os próximos três elementos como argumentos. Esses argumentos representam, respectivamente, o endereço do primeiro inteiro da multiplicação, o endereço do segundo inteiro da multiplicação, e o endereço onde deve ser escrito o resultado do produto. Após esse processo, o ponteiro de execução se encontra no elemento seguinte ao último argumento. Veja a tabela contida no manual:

Operação 2: MUL											
	0	1	2	3	4	5	6	7	8	9	10
Antes	2	5	6	7	0	3	5	4	0	0	-1
<code>mul <add1> <add2> <dest></code>											
	0	1	2	3	4	5	6	7	8	9	10
Depois	2	5	6	7	0	3	5	15	0	0	-1
Atribuir ao endereço <dest> o valor do produto dos conteúdos de <add1> e <add2>											

1.4 Operação 3 - CLT (COMPARE LESS THAN)

Essa é uma operação de comparação entre dois inteiros escritos em endereços (índices começando em 0) da fita, representada pelo inteiro 3. Quando um 3 é lido como operação, o interpretador deve ler os próximos três elementos como argumentos. Esses argumentos representam, respectivamente, o endereço do primeiro inteiro da comparação, o endereço do segundo inteiro da comparação, e o endereço onde deve ser escrito o resultado dessa comparação. Se o primeiro inteiro for menor que o segundo tem-se 1 (true) como resultado, caso contrário o resultado será 0 (false). Após esse processo, o ponteiro de execução se encontra no elemento seguinte ao último argumento. Veja a tabela contida no manual:

Operação 3: CLT											
	0	1	2	3	4	5	6	7	8	9	10
Antes	3	5	6	7	0	3	5	4	0	0	0
<code>clt <add1> <add2> <dest></code>											
	0	1	2	3	4	5	6	7	8	9	10
Depois	3	5	6	7	0	3	5	1	0	0	0
Atribuir ao endereço <dest> 1 se <add1> for menor que <add2>, e 0 se o contrário											

1.5 Operação 4 - CEQ (COMPARE EQUALS)

Essa é uma operação de comparação entre dois inteiros escritos em endereços (índices começando em 0) da fita, representada pelo inteiro 4. Quando um 4 é lido como operação, o interpretador deve ler os próximos três elementos como argumentos. Esses argumentos representam, respectivamente, o endereço do primeiro inteiro da comparação, o endereço do segundo inteiro da comparação, e o endereço onde deve ser escrito o resultado dessa comparação. Se o primeiro inteiro igual ao segundo tem-se 1 (true) como resultado, caso contrário o resultado será 0 (false). Após esse processo, o ponteiro de execução se encontra no elemento seguinte ao último argumento. Veja a tabela contida no manual:

Operação 4: CEQ											
	0	1	2	3	4	5	6	7	8	9	10
Antes	4	5	6	7	0	3	5	4	0	0	0
<code>ceq <add1> <add2> <dest></code>											
	0	1	2	3	4	5	6	7	8	9	10
Depois	4	5	6	7	0	3	5	0	0	0	0
Atribuir ao endereço <dest> 1 se <add1> for igual a <add2>, e 0 se o contrário											

1.6 Operação 5 - JMP (JUMP)

Essa é uma operação de alteração do ponteiro de execução, representada pelo inteiro 5. Quando um 5 é lido como operação, o interpretador deve ler o próximo elemento como argumento. Esse argumento representa o endereço onde está contido o próximo valor para ponteiro de execução, ou seja o endereço a partir de onde o programa deve continuar. Após esse processo, o ponteiro de execução se encontra no elemento para o qual ele pulou. Veja a tabela contida no manual:

Operação 5: JMP																																		
<table border="1"> <tr> <td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td></tr> <tr> <td>Antes</td><td>5</td><td>5</td><td>6</td><td>7</td><td>0</td><td>3</td><td>5</td><td>4</td><td>0</td><td>0</td><td>0</td></tr> </table>												0	1	2	3	4	5	6	7	8	9	10	Antes	5	5	6	7	0	3	5	4	0	0	0
0	1	2	3	4	5	6	7	8	9	10																								
Antes	5	5	6	7	0	3	5	4	0	0	0																							
<i>jmp <add1></i>																																		
<table border="1"> <tr> <td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td></tr> <tr> <td>Depois</td><td>5</td><td>5</td><td>6</td><td>7</td><td>0</td><td>3</td><td>5</td><td>4</td><td>0</td><td>0</td><td>0</td></tr> </table>												0	1	2	3	4	5	6	7	8	9	10	Depois	5	5	6	7	0	3	5	4	0	0	0
0	1	2	3	4	5	6	7	8	9	10																								
Depois	5	5	6	7	0	3	5	4	0	0	0																							
Ir para o endereço armazenado no conteúdo de <add1>																																		

1.7 Operação 6 - JEQ (JUMP IF EQUAL)

Essa é uma operação de alteração condicional do ponteiro de execução, representada pelo inteiro 6. Quando um 6 é lido como operação, o interpretador deve ler os dois próximos elementos como argumentos. Esses argumentos representam, respectivamente, o endereço (índice começando em 0) do valor para a condicional, e o endereço onde está contido o próximo valor para ponteiro de execução, ou seja o endereço a partir de onde o programa deve continuar caso a condição seja atendida, senão o ponteiro mantém seu incremento normalmente. A condição é atendida se o valor da condicional for diferente de 0 (false). Após esse processo, o ponteiro de execução se encontra no elemento para o qual ele pulou quando a condição for atendida, e, caso contrário, no elemento seguinte ao último argumento. Veja a tabela contida no manual:

Operação 6: JEQ																																		
<table border="1"> <tr> <td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td></tr> <tr> <td>Antes</td><td>6</td><td>5</td><td>6</td><td>7</td><td>0</td><td>3</td><td>4</td><td>4</td><td>0</td><td>0</td><td>0</td></tr> </table>												0	1	2	3	4	5	6	7	8	9	10	Antes	6	5	6	7	0	3	4	4	0	0	0
0	1	2	3	4	5	6	7	8	9	10																								
Antes	6	5	6	7	0	3	4	4	0	0	0																							
<i>jeq <add1> <add2></i>																																		
<table border="1"> <tr> <td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td></tr> <tr> <td>Depois</td><td>6</td><td>5</td><td>6</td><td>7</td><td>0</td><td>3 (!= 0)</td><td>4</td><td>4</td><td>0</td><td>0</td><td>0</td></tr> </table>												0	1	2	3	4	5	6	7	8	9	10	Depois	6	5	6	7	0	3 (!= 0)	4	4	0	0	0
0	1	2	3	4	5	6	7	8	9	10																								
Depois	6	5	6	7	0	3 (!= 0)	4	4	0	0	0																							
Se o conteúdo de <add1> for diferente de 0, ir para o endereço no conteúdo de <add2>																																		

1.8 Operação 7 - CPY (COPY)

Essa é uma operação de cópia entre dois endereços (índices começando em 0) da fita, representada pelo inteiro 7. Quando um 7 é lido como operação, o interpretador deve ler os próximos dois elementos como argumentos. Esses argumentos representam, respectivamente, o endereço de origem para cópia, e o endereço de destino para cópia. O valor contido no endereço de origem deve ser copiado para o endereço de destino. Após a execução, o ponteiro de execução se encontra no elemento seguinte ao último argumento. Veja a tabela contida no manual:

Operação 7: CPY											
	0	1	2	3	4	5	6	7	8	9	10
Antes	7	5	6	7	0	3	5	4	0	0	0
<code>cpy <add1> <dest></code>											
Depois	7	5	6	7	0	3	3	4	0	0	0
Copia para o endereço <dest> o conteúdo de <add1>											

1.9 Operação 8 - PUT

Essa é uma operação de impressão de um caractere na saída, representada pelo inteiro 8. Quando um 8 é lido como operação, o interpretador deve ler o próximo elemento como argumento. Esse argumento representa o endereço (índice começando em 0) do valor — representante inteiro ASCII — do caractere que deve ser printado. Após a execução, o ponteiro de execução se encontra no elemento seguinte ao argumento. Veja a tabela contida no manual:

Operação 8: PUT											
	0	1	2	3	4	5	6	7	8	9	10
Antes	8	5	6	7	0	65	5	4	0	0	0
<code>put <add1></code>											
Depois	8	5	6	7	0	65	5	4	0	0	0
Imprime o caractere correspondente ao inteiro na tabela ASCII, do conteúdo do endereço <add1>, neste caso ele mostraria o caractere 'A'											

1.10 Operação 9 - PTN (PUT NUM)

Essa é uma operação de impressão de um inteiro na saída, representada pelo inteiro 9. Quando um 9 é lido como operação, o interpretador deve ler o próximo elemento como argumento. Esse argumento representa o endereço (índice começando em 0) do valor inteiro que deve ser printado. Após a execução, o ponteiro de execução se encontra no elemento seguinte ao argumento. Veja a tabela contida no manual:

Operação 9: PTN											
	0	1	2	3	4	5	6	7	8	9	10
Antes	9	5	6	7	0	65	5	4	0	0	0
<i>ptn <add1></i>											
Depois	9	5	6	7	0	65	5	4	0	0	0
Imprime o inteiro do conteúdo do endereço <i><add1></i> , neste caso ele mostraria o inteiro 65											

Cleyton já ouviu diversas vezes todas as fitas de punk deixadas pelo seu instrutor, isto é, já passaram alguns anos, porém a barata ainda não entendeu muito bem como fará para construir um interpretador da linguagem tape, resta pouco tempo, e existe um problema ainda maior: O Pupilo Punk simplesmente não consegue digitar nos teclados feitos pelos seres humanos, e Supla não conseguiu criar um teclado adaptado para seu aprendiz.

Por sorte, enquanto se lamentava nos laboratórios do ICMC, Cleyton notou a presença de outro ser no local, um humano jogado em cima de um computador qualquer. Misteriosamente o humano era resistente a radiação e não estava morto, mas sim dormindo. E barata gritou: "Boto fé que é tu, o destino trouxe alguém pra ajudar, acorda aê, bo!".

O humano era você, o escolhido por Cleyton para ajudar na salvação da humanidade em 2077. Escreva um programa em C que, dado uma fita representante de um programa na linguagem tape, execute esse programa mostrando a saída esperada por ele e o estado final da fita

2 Entrada

Tem-se como entrada uma sequência de 512 inteiros, separados linha a linha, representantes de uma fita executável na linguagem tape, um programa válido (sem operações inválidas) de acordo com as especificações da linguagem.

É válido ressaltar que devem ser usados os recursos vistos até então nessa matéria (operadores, condicionais, loops, e vetores).

3 Saída

Dada a fita de entrada, interprete e execute o programa contido na fita, imprimindo a saída dele caso exista, e, ao final da execução, imprima o estado final da fita (todos 512 elementos linha a linha), com suas alterações. Separe a saída do programa em tape e o estado final da fita com uma linha vazia.

4 Exemplos de entrada e saída

Entrada genérica

```
1 <primeiro elemento da fita>
2 <segundo elemento da fita>
3 <terceiro elemento da fita>
4 <quarto elemento da fita>
5 .
6 .
7 .
8 <penultimo elemento da fita>
9 <ultimo elemento da fita>
```

Saída genérica

```
1 Saída do programa:
2 <saída do programa executado em tapec>
3
4 Estado final da fita:
5 <primeiro elemento da fita final>
6 <segundo elemento da fita final>
7 .
8 .
9 .
10 <penultimo elemento da fita final>
11 <ultimo elemento da fita final>
```

Os exemplos de entradas e saídas reais podem ser acessados por esta [pasta do drive](#) ou mesmo via download do .zip com os casos testes no [runcodes](#).

Bom trabalho :)