

O problema da parada

O Problema da Parada (The Halting Problem) é um problema de decisão clássico da Ciência da Computação que consiste, basicamente, em determinar se um dado programa sempre vai parar (ou seja, terminar sua execução) para uma dada entrada arbitrária ou se vai executar infinitamente. Alan Turing provou, em 1936, que é impossível resolver o problema da parada generalizando para qualquer par programa-entrada. Neste problema, porém, dada a descrição de uma linguagem simples, um programa escrito nessa linguagem e uma entrada para esse programa, você deve determinar se o programa dado pára com a entrada dada e, em caso positivo, qual a saída produzida.

Esta linguagem só trabalha com números inteiros de 0 a 999 (inclusive). Sendo assim, o sucessor de 999 é 0, e o antecessor de 0 é 999. Além disso, ela possui dez variáveis (R0 a R9), sendo que a R0 sempre é atribuído o valor de chamada do programa (ou seja, o parâmetro de entrada) e a R9 é sempre atribuído o valor de saída (o retorno). No início da execução do programa, é atribuído o valor 0 a todas as variáveis, com exceção de R0 que recebe o parâmetro de entrada.

As operações básicas são atribuição (MOV), soma (ADD), subtração (SUB), multiplicação (MUL), divisão inteira (DIV) e resto da divisão inteira (MOD). Todas essas operações têm a sintaxe `COMANDO OPERANDO1,OPERANDO2` (sem espaços entre a vírgula e os operandos), onde `COMANDO` é uma dessas operações, `OPERANDO1` é uma das 10 variáveis (R0 a R9) e `OPERANDO2` pode ser uma das 10 variáveis ou um valor inteiro (entre 0 e 999). Todas as operações modificam o valor de `OPERANDO1`, sendo assim `MOV R4,100` é o equivalente a atribuir o valor 100 a R4, enquanto que `MUL R3,R8` é o equivalente a multiplicar R3 por R8 e atribuir o resultado a R3. A operação `DIV`, assim como a `MOD`, retornam 0 (zero) se `OPERANDO2` for 0 ou se a variável equivalente tiver valor 0. Ou seja, `DIV R4,0` é o equivalente a `MOV R4,0`. Por divisão inteira, entendemos a parte inteira do quociente da divisão (sem a parte fracionária). Por exemplo, a divisão inteira de 7 por 2 é 3 (sendo o resto 1).

Existem seis comandos de fluxo de decisão: `IFEQ` (se igual), `IFNEQ` (se diferente), `IFG` (se maior), `IFL` (se menor), `IFGE` (se maior ou igual) e `IFLE` (se menor ou igual). A sintaxe de todos eles é `COMANDO OPERANDO1,OPERANDO2` (sem espaços entre a vírgula e os operandos), onde `OPERANDO1` e `OPERANDO2` podem ser variáveis (R0 a R9) ou valores inteiros (entre 0 e 999). Assim, o comando `IFEQ R4,123` é o equivalente a testar se R4 é igual a 123. Caso a condição testada seja verdadeira, o programa continua a executar normalmente a linha subsequente ao comando de decisão. Caso a condição seja falsa, o programa passa a executar a linha subsequente ao `ENDIF` mais próximo. Todos os comandos de decisão devem ter um comando `ENDIF` correspondente.

Finalmente, existem os comandos `CALL` e `RET`, ambos com a sintaxe `COMANDO OPERANDO`, onde `OPERANDO` é uma variável (R0..R9) ou valor direto (entre 0 e 999). O comando `CALL` chama o próprio programa novamente, passando `OPERANDO` como parâmetro de entrada, ou seja, atribuindo o valor de `OPERANDO` à variável R0. Já `RET` termina a execução do programa, retornando o valor de `OPERANDO` como o resultado de saída. A última linha do programa sempre será um comando `RET`. Observe que, caso o programa chame a si mesmo através do comando `CALL`, quando a execução voltar, o valor de R9 vai estar alterado com o valor retornado pelo programa. Note também que todas as variáveis (R0..R9) são locais, ou seja, uma

chamada subsequente ao programa não pode alterar os valores guardados nas variáveis da instância anterior, com exceção, naturalmente, do valor de R9 que recebe o retorno da instância chamada.

O exemplo a seguir ilustra um programa que calcula o fatorial de um número.

```
linha comando
1  IFEQ R0,0
2   RET 1
3  ENDIF
4  MOV R1,R0
5  SUB R1,1
6  CALL R1
7  MOV R2,R9
8  MUL R2,R0
9  RET R2
```

1a linha: Verifica se o valor de R0 vale 0, caso positivo, executa a próxima linha, caso contrário, pula para a 4a linha (ENDIF mais próximo).

2a linha: Retorna 1 como saída do programa.

3a linha: Marca o fim do bloco de decisão iniciado na primeira linha.

4a linha: Atribui o valor de R0 a R1 ($R1 \leftarrow R0$).

5a linha: Diminui 1 de R1 ($R1 \leftarrow R1 - 1$).

6a linha: Chama o programa passando R1 como parâmetro de entrada.

7a linha: Guarda o valor de R9 (retornado pela chamada anterior) em R2 ($R2 \leftarrow R9$)

8a linha: Multiplica o valor de R2 por R0 ($R2 \leftarrow R2 * R0$)

9a linha: Retorna o valor de R2 como saída do programa.

A tabela a seguir traz um resumo dos comandos para referência:

comando	sintaxe	significado
MOV	MOV OP1,OP2	$OP1 \leftarrow OP2$
ADD	ADD OP1,OP2	$OP1 \leftarrow OP1 + OP2$
SUB	SUB OP1,OP2	$OP1 \leftarrow OP1 - OP2$
MUL	MUL OP1,OP2	$OP1 \leftarrow OP1 * OP2$
DIV	DIV OP1,OP2	$OP1 \leftarrow OP1 / OP2$
MOD	MOD OP1,OP2	$OP1 \leftarrow OP1 \% OP2$
IFEQ	IFEQ OP1,OP2	if $OP1 == OP2$
IFNEQ	IFNEQ OP1,OP2	if $OP1 != OP2$
IFG	IFG OP1,OP2	if $OP1 > OP2$
IFL	IFL OP1,OP2	if $OP1 < OP2$
IFGE	IFGE OP1,OP2	if $OP1 \geq OP2$
IFLE	IFLE OP1,OP2	if $OP1 \leq OP2$
ENDIF	ENDIF	Marca fim do bloco de execução condicional
CALL	CALL OP	Chama o programa com OP como entrada
RET	RET OP	return OP

Entrada

A entrada contém vários casos de teste. Cada caso de teste se inicia com dois inteiros, L e N , representando respectivamente o número de linhas do programa ($1 \leq L < 100$) e o valor do

parâmetro de entrada do programa ($0 \leq N < 1000$). As L linhas seguintes contêm o programa. Pode-se assumir que ele está sempre sintaticamente correto de acordo com as regras definidas acima. Todos os comandos (bem como o nome das variáveis) só conterão letras maiúsculas. O final da entrada é marcado pelo caso em que $L = N = 0$ e não deve ser processado.

Saída

Para cada caso de teste, seu programa deve produzir uma linha contendo um inteiro que representa o valor de saída (retorno) para a entrada N dada, ou um asterisco (*) no caso de o programa nunca terminar.

Exemplo de entrada

```
9 6
IFEQ R0,0
RET 1
ENDIF
MOV R1,R0
SUB R1,1
CALL R1
MOV R2,R9
MUL R2,R0
RET R2
2 123
CALL R0
RET R0
0 0
```

Saída para o exemplo de entrada

```
720
*
```