Capítulo 1

Computadores Hipotéticos

Neste Capítulo, serão apresentados três modelos de computadores hipotéticos : HV-1, HV-2 e HIPO. O estudo do funcionamento desses modelos tem como objetivo providenciar explicações facilitadas por meio de analogias sobre os pormenores dos computadores reais. Esse estudo também tem como objetivo o aprendizado de conceitos fundamentais da programação de computadores, tais como os conceitos de variável e programa armazenado.

Ao indivíduo que utiliza os computadores hipotéticos denominaremos por **usuário**. Este usuário dispondo de um conjunto de instruções, que representem a solução computacional de algum problema, as insere no computador por algum **dispositivo de entrada** e dá um comando de início de execução. O computador hipotético passa a processar as instruções, sem qualquer interferência do usuário até que exista uma instrução que requeira interferência do usuário (entrada de dados), ou ocorra algum evento inesperado (como uma divisão por zero), ou termine a execução das instruções.

1.1 O Computador HV-1

O computador HV-1 é formado pelos seguintes componentes :

1. Um gaveteiro com 100 gavetas;
2. Uma calculadora com mostrador e teclado;
3. Um pequeno quadro-negro denominado EPI;
4. Um porta-cartões;
5. Uma folha de saída; e
6. Um operador do sistema, uma pessoa chamada CHICO, com lápis, apagador de quadro-negro e giz.

1.1.1 Gaveteiro

O gaveteiro consiste numa sequência de gavetas numeradas de 00 a 99. O número de cada gaveta é denominado seu endereço. Cada gaveta contém pequeno quadro-negro, onde é escrito um número sempre com 3 algarismos (por exemplo, 102, 003, etc.) e outras informações que veremos mais tarde. O gaveteiro construído de tal maneira que valem as seguintes regras de utilização:

1. em qualquer momento, no máximo uma gaveta pode estar aberta;
2. a leitura do quadro-negro de uma gaveta não altera o que nele está escrito (gravado);

3. a escrita de uma informação no quadro-negro de uma gaveta ´e sempre precedida do apagamento do mesmo; e

4. somente o operador CHICO tem acesso ao gaveteiro.

1.1.2 Calculadora

Trata-se de uma calculadora usual, com teclado para entrada de números, teclas das quatro operações aritméticas básicas, tecla '=' e um mostrador, que denominaremos acumulador. Não há tecla de ponto (ou vírgula) decimal ou outra tecla adicional qualquer. Há dois tipos de operações efetuadas com essa calculadora:

1. carga de um número no acumulador. Para isso, pressiona-se a tecla '=' (garantindo-se assim o encerramento de alguma operação prévia) e a seguir "digitam-se" os algarismos do número a ser carregado, o qual aparece no acumulador;

2. operação aritmética. Esta é sempre feita entre o número que está no acumulador e um segundo número. Para isso, pressiona-se a tecla da operação desejada, digita-se o segundo número e pressiona-se a tecla '='. O resultado da operação aparece no acumulador.

Assim como o gaveteiro, a calculadora só pode ser utilizada pelo CHICO.

1.1.3 EPI

Trata-se de quadro-negro independente do gaveteiro, com a forma  , onde será escrito um número entre 00 e 99, correspondendo a um endereço de gaveta do gaveteiro. O número nele escrito indica sempre o "Endereço da Próxima Instrução", donde sua abreviatura. O conceito de "instrução" será explicado adiante.

Somente o CHICO tem acesso ao EPI.

1.1.4 Porta-Cartões

É um dispositivo similar aos porta-cigarros onde são empilhados maços de cigarro a serem vendidos. O porta-cartões funciona de acordo com as seguintes regras:

1. cartões com informações são colocados exclusivamente pela parte superior, um a um; quando um cartão contém um número, este é sempre escrito com 3 algarismos, como por exemplo, 101, 003, etc;

2. cartões são retirados da extremidade inferior, um de cada vez, aparecendo na mesma ordem em que foram colocados no dispositivo;

3. a retirada de cartões só pode ser feita pelo CHICO; e

4. a colocação de cartões só pode ser feita pelo usuário.

1.1.5 Folha de Saída

Trata-se de uma folha de papel onde pode ser escrito um número em cada linha,utilizando-se sempre linhas consecutivas. Somente o CHICO pode escrever nessa folha; somente o usuário pode ler o que já foi escrito.

1.1.6 Operador

Resumindo as diversas características descritas, vemos que o operador CHICO é a única pessoa que tem acesso ao gaveteiro, à calculadora, ao EPI, e é o único que pode retirar cartões do porta-cartões e escrever na folha de saída. Ele executa estritamente ordens recebidas, não podendo tomar nenhuma iniciativa própria, executando alguma ação fora da especificação dessas ordens.

O CHICO trabalha sempre em um de dois estados diferentes:

1. **Estado de carga**, onde ele exclusivamente transcreve informações de cartões, lidos do porta-cartões, para gavetas do gaveteiro.

2. Estado de execução, onde ele executa ordens gravadas nas gavetas. Os detalhes do funcionamento desses estados ser˜ao explicados adiante.

A comunicação entre o usuário e o operador é feita exclusivamente através das unidades porta-cartões e folha de saída. O CHICO sabe fazer "de cabeça" uma única operação aritmética: incrementar de 1 o conteúdo do EPI.

1.1.7 Programando o HV-1

Para resolver um problema usando o computador HV-1,o usuário deve planejar uma sequência de ordens (o programa) a serem executadas pelo CHICO. Cada uma dessas ordens é denominada instrução. Um exemplo de instrução é o seguinte: "some o conteúdo da gaveta de endereço 41 ao conteúdo do acumulador". A fim de se produzir a execução correta das instruções e na sequência adequada, elas são escritas nas gavetas do gaveteiro. Para executar uma instrução da sequência, o CHICO segue os seguintes passos:

1. consulta o EPI, onde está escrito o endereço E da próxima instrução;

2. incrementa de 1 o conteúdo do EPI, apagando o valor anterior e escrevendo o novo valor (o qual neste caso será E+1);

3. abre a gaveta de endereço E; nesta gaveta ele deve encontrar uma instrução I, que é lida;

4. fecha a gaveta E; e

5. executa I.

Após finalizados esses passos, o CHICO recomeça do passo 1, com exceção de um caso explicado a seguir. Se a execução da instrução I não acarretar alteração no conteúdo do EPI, a próxima instrução a ser executada será a da gaveta de endereço E+1, devido ao passo 2. Se uma instrução acarretar alteração no EPI, mudando o seu conteúdo para X, a próxima instrução a ser executada será a da gaveta de endereço X; diz-se que houve um desvio para X.

As instruções escritas nas gavetas do gaveteiro constituem um programa armazenado. Para conseguir a execução de um programa, o usuário deve produzir, inicialmente, o armazenamento desse programa no gaveteiro, passando portanto a constituir um programa armazenado. Isso é feito da seguinte forma:

1. o usuário escreve cada instrução em um cartão, precedida de um endereço; assim, cada cartão do programa contém um par ordenado (E,I), onde E é um endereço e I uma instrução;

2. o CHICO é colocado em estado de carga de programa;

3. o usuário coloca o conjunto de cartões do programa no porta-cartões, em qualquer ordem;

4. como o CHICO está em estado de carga, ele lê um cartão com um par (E,I); abre a gaveta de endereço E; escreve em seu quadro-negro a instrução I; fecha essa gaveta;

5. o CHICO repete o passo 4 até ler o último cartão de programa, após o que ele é colocado em **estado de execução de programa**.

Após o encerramento da carga do programa, o CHICO é colocado em estado de execução de programa. Isso é feito por meio de um cartão especial, que deve encerrar o conjunto de cartões de programa. A forma desse cartão é "EXECUTE X", onde X é um número escrito pelo usuário; será o endereço da gaveta onde se encontra a primeira instrução a ser executada.

Ao ler esse cartão, o CHICO apaga o EPI e escreve o mesmo valor X; a seguir, ele vai para o passo 1 da execução de uma instrução, como exposto no início deste item.

Para completar este quadro, resta descrever como o CHICO entra em estado de carga de programa. Vamos supor que, na verdade, esse estado seja o estado "normal" do CHICO; ele só pode sair desse estado ao tentar carregar um cartão "EXECUTE X". Estando em estado de execução, ele só sai desse estado nos dois casos seguintes:

1. através da execução da instrução "pare a execução";

2. se ocorrer algum erro durante a execução. Um exemplo do caso 2 é o de o CHICO tentar executar uma instrução inválida, isto é, não conhecida.

1.1.8 Instruções do HV-1

O conteúdo de uma gaveta de endereço E, isto é, o número gravado em seu quadro-negro, será representado por cE. Assim, c10 indicará o conteúdo da gaveta 10. Indicaremos por cAC o conteúdo do acumulador; este será abreviado por AC.

Soma

- Instrução: "some o cE ao AC".
- Significado: some o cE ao cAC e coloque o resultado no AC; o cE não se altera.
- Execução: o CHICO efetua os seguintes passos:
    1. digita a tecla '+' da calculadora;
    2. abre a gaveta de endereço E;
    3. lê o número escrito nessa gaveta (cE) e digita-o na calculadora;
    4. fecha a gaveta E; e
    5. digita '=' na calculadora.

Daqui em diante, em lugar de escrevermos "gaveta de endereço E", escreveremos simplesmente E. Também deixaremos de mencionar explicitamente que é o CHICO quem efetua os passos da execução.

Subtração

- Instrução: "subtraia o cE de AC".
- Significado: subtraia o cE de cAC e coloque o resultado no AC; o cE não se altera.
- Execução: o CHICO efetua os seguintes passos:
    1. digita a tecla '-' da calculadora;

2. abre a gaveta de endereço E;
3. lê o número escrito nessa gaveta (cE) e digita-o na calculadora;
4. fecha a gaveta E; e
5. digita '=' na calculadora.

Carga no AC

- Instrução: "carregue o cE no AC".
- Significado: copie o cE no AC; o cE não muda.
- Execução:
  1. digita '=';
  2. abre E;
  3. lê cE e digita-o; e
  4. fecha E.

Armazenamento do AC

- Instrução: "armazene o cAC em E".
- Significado: copie o cAC em E; o cAC não muda (oposto da instrução anterior).
- Execução:
  1. abre E;
  2. apaga o cE;
  3. lê o cAC e o escreve em E; e
  4. fecha a gaveta.

Impressão

- Instrução: "imprima o cE".
- Significado: o cE é transcrito na folha de saída.
- Execução:
  1. abre E;
  2. lê cE e escreve seu valor na próxima linha da folha de saída; e
  3. fecha a gaveta.

Note que supusemos haver espaço na folha de saída. No caso contrário, o CHICO aguarda até ser fornecida nova folha.

Leitura

- Instrução: "leia um cartão e guarde em E".
- Significado: o conteúdo do próximo cartão do porta-cartões é lido e transcrito para E;
- Execução:
  1. abre E;
  2. retira um cartão do porta-cartões;
  3. lê o conteúdo do cartão e escreve o seu valor em E;
  4. joga fora o cartão; e
  5. fecha E.

Note que supusemos haver cartão no porta-cartões. Em caso contrário, o CHICO aguarda a colocação de pelo menos um cartão no porta-cartões.

Desvio Condicional

- Instrução: "se cAC$\neq$0, desvie para E".
- Significado: se há um número diferente de 0 no AC, a próxima instrução a ser executada está em E, caso contrário não há nada a fazer (isto é, a próxima instrução a ser executada estará na gaveta seguinte à que contém esta instrução).
- Execução:
  1. lê o cAC; e
  2. se cAC$\neq$0 então apaga o EPI e escreve E no mesmo.

Pare

- Instrução: "pare".
- Significado: encerra a execução do programa.
- Execução:
  1. entrega a folha de saída para o usuário; e
  2. entra no estado de carga.

1.2 Estudo de Caso no Computador HV-1

Nesse capítulo, o modelo do Computador HV-1, estabelecido no capítulo anterior, será utilizado para ilustrar o funcionamento de um computador durante a resolução de problemas computacionais, e como suporte para o aprendizado de conceitos fundamentais de algoritmos e programação.

De maneira concomitante, o processo de resolução será detalhado de modo a providenciar um método de abordagem de problemas computacionais que possa ser replicável, e que diminua a complexidade do processo de resolução de tais problemas.

1.2.1 Como resolver problemas com o Computador HV-1

A resolução de problemas computacionais é um processo que pode ser extenuante do ponto de vista cognitivo, pois, de acordo com Michal Armoni e Liat Nakar [1], tais tarefas dependem fundamentalmente do conceito de abstração, definido de maneira geral como uma capacidade de ignorar detalhes irrelevantes de acordo com o contexto.

Como o contexto sofre mudanças durante o processo, é necessário transitar entre diferentes níveis de abstração e detalhes para de fato resolver um problema computacional.

Para modelar o ensino de programação e explicar como a abstração se estrutura durante o processo de resolução de problemas computacionais, foram desenvolvidos o ATT (Abstraction Transition Taxonomy) e o framework LOA (Levels of abstraction) [4].

ATT e objetivos do ensino de programação

A taxonomia de transição de abstração (ATT) é uma taxonomia de ensino com o objetivo de classificar os níveis de abstração envolvidos na programação e providenciar linguagem comum para discussão de temas relacionados ao ensino de programação. Essa taxonomia relaciona semelhanças linguísticas com diferentes níveis de abstração, sendo eles : Português, Pseudocódigo e Código. [3]

Como parte do processo de criação da taxonomia, também foram identificados que os problemas computacionais são divididos em dois grandes grupos : O primeiro grupo de problemas investiga definições e motivos; O segundo grupo investiga como implementar as resoluções por meio dos programas. [3]

Por fim esses dois grupos de problemas e os três níveis de abstração compõem uma série transições que podem surgir durante o processo de resolução de problemas computacionais e dominar essas transições é o objetivo do ensino de algoritmos e programação. [3]

LOA e a abstração como conceito central

De maneira semelhante, o framework LOA trata a programação, enquanto produção de código, como parte de um processo maior de resolução de problemas, ligado ao conceito de abstração. No entanto, esse framework define os níveis de detalhes em quatro : Problema, Algoritmo, Programa e Execução. [2].

Cada nível de abstração apresenta suas especificidades e seus focos, sendo eles :

- Nível 4 - Problema : Trata-se da primeira etapa de resolução de problemas computacionais e consiste na análise conceitual do problema a ser resolvido, focando no que deve ser resolvido e quais são as limitações impostas para resolução.
- Nível 3 - Algoritmo : Consiste na primeira fase de análise de uma possível resolução para o problema, contudo, o foco dessa fase é providenciar etapas para a resolução do problema de modo que tal resolução seja generalizada, ignorando detalhes de implementação.
- Nível 2 - Programa : Esta etapa é uma aplicação do algoritmo ao contexto específico, ou seja, uma implementação, e considera detalhes técnicos e comandos disponíveis. Caso a implementação correta do algoritmo não resolva o problema, é necessário voltar ao nível de algoritmo para elaborar uma nova solução antes de implementar um novo programa.
- Nível 1 - Execução : Última etapa do desenvolvimento e consiste uma determinada execução de um programa com determinadas entradas em uma máquina específica. [2].

Assim, o processo de resolução de problemas, de acordo com a utilização do LOA, é feito começando pelo nível de problema e transicionando entre os níveis de abstração quando necessário. Abordar a abstração em níveis possibilita diminuir a quantidade de informação gerenciada em cada passo do processo de modo a facilitar a resolução de problemas. [2]

Aplicando os conceitos ao HV-1

Dessa maneira, os modelos de aprendizado de computação serão utilizados em conjunto com o Computador HV-1 para providenciar uma estrutura para resolução de problemas computacionais com menor carga cognitiva e para o desenvolvimento de conceitos de algoritmo e programação, incluindo a abstração nos seus diferentes níveis.

Como apresentando anteriormente na explicação dos modelos ensino, a resolução de problemas nesse capítulo vai adotar quatro etapas : Problema, Algoritmo, Programa e Execução, considerando os diferentes conjuntos linguísticos associados (Português, Pseudocódigo e Código). É recomendado que o estudante adote essas etapas explicitamente em seu processo de resolução de problemas computacionais para melhor gerenciar a dificuldade das soluções e exercitar sua capacidade de abstração.

Para ilustrar esse processo, as próximas seções apresentam alguns exemplos de problemas e seus respectivos processos de solução e implementação no HV-1.

1.2.2 Somando uma sequência de números ($N = 2$)

Problema : *São dados dois números inteiros, determine sua soma utilizando o Computador HV-1 e retorne o resultado para o usuário.*

Problema : Somando Dois Números

Como foi estabelecido anteriormente, a abordagem dos problemas descritos nesse estudo de caso serão estruturados em quatro etapas, sendo a primeira delas a abordagem conceitual do problema, ou seja, análise da lógica por trás do problema visando entender como ele poder ser resolvido, sem se preocupar com a formalização lógica tampouco detalhes a nível de implementação.

O problema a ser resolvido consiste em realizar uma operação de soma entre dois números inteiros, que serão inseridos pelo usuário, onde o resultado deve ser disponibilizado de volta ao usuário. Como o problema não coloca nenhuma restrição adicional, é possível prosseguir para o planejamento da solução, ou seja, o algoritmo referente ao problema da soma de dois números inteiros.

Algoritmo : Somando Dois Números

Após a análise conceitual do problema, é necessário delinear uma solução possível para resolver o problema, que, no caso, consiste na soma de dois números inteiros. É importante que na etapa de criação do algoritmo seja dada ênfase à idealização de uma solução geral que não se preocupa com detalhes de implementação, ou seja, com código de maneira geral.

Para resolver o problema da soma de dois números inteiros, torna-se necessário dispor dos dois números inteiros e utilizá-los para a operação de soma, coletando seu resultado e retornando ao usuário como foi solicitado. Uma representação visual é disposta na Figura 1.1.

Perceba que na etapa de algoritmo, a resolução do problema começa descolada do seu ambiente de implementação, onde o código será escrito, deixando algumas dúvidas mais específicas sobre como será resolvido o problema. Caso a etapa da programação imponha restrições à solução possível, é necessário voltar à etapa de elaboração dos algoritmos para idealizar uma nova forma de resolver o problema.

Indo além do escopo do algoritmo, temos que, no contexto do Computador HV-1, receber os números significa enviá-los ao Chico via porta-cartões durante a execução do programa e, dessa forma, Chico pode armazená-los para efetuar a operação de soma. Por fim, retornar o resultado ao usuário significa que Chico escreveu o valor da soma na folha de saída.

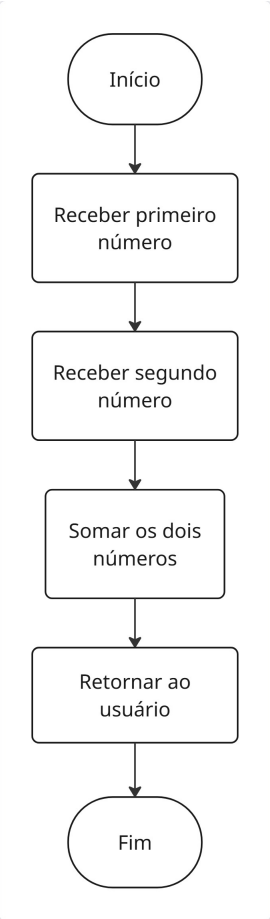


Figura 1.1: Algoritmo para o problema da soma de dois números. Fonte : Elaborado pelos autores.

Programa : Somando Dois Números

Em continuidade à etapa algoritmo, se situa a etapa do programa, ou seja, nível de programação da solução disposta no algoritmo da soma de dois números, na plataforma HV-1, de modo a resolver o problema. Para implementar uma solução de problema computacional é necessário considerar as operações disponíveis e a sintaxe da plataforma de implementação, que no caso, é o Computador HV-1.

Assim, se faz necessário entender como implementar os procedimentos de "Receber um número", "Somar dois números"e "Retornar ao usuário"na plataforma HV-1 com base em seus comandos disponíveis e de modo a materializar a solução.

Aqui entende-se procedimento como um bloco composto por uma ou mais instruções, que apresentam um certo nível de coesão e que são responsáveis por implementar uma sub-tarefa do algoritmo ao qual o programa se refere.

Procedimento: Receber um número

O primeiro procedimento necessário para implementar a solução disposta na seção anterior é receber os dois números envolvidos na soma. Esse procedimento apresenta uma instrução específica no ambiente do Computador HV-1 que é "leia um cartão e guarde em E", onde E é um número que representa uma das gavetas do computador.

Perceba que no ambiente HV-1, receber um valor é indissociável de seu armazenamento, para que o valor fique acessível ao Chico.

Procedimento: Somar dois números

Em seguida, deve-se realizar a operação de soma dos números. Como no procedimento anterior, a soma apresenta uma instrução nativa do Computador HV-1 que é "some o cE ao AC", onde cE é o conteúdo da gaveta de número E e AC é o acumulador.

Perceba que o Computador HV-1 associa a operação de soma ao acumulador, o que faz com que seja necessário colocar um dos valores envolvidos na soma no acumulador antes de executar a instrução citada. Depois de executar a operação, se faz necessário armazenar o resultado em alguma gaveta, para que Chico seja capaz de utilizar o resultado em algum outro procedimento.

Considerando que tanto a carga quanto o armazenamento tem instruções próprias, a Figura 1.2 demonstra um possível procedimento de soma, supondo que os números estão guardados nas gavetas A e B.

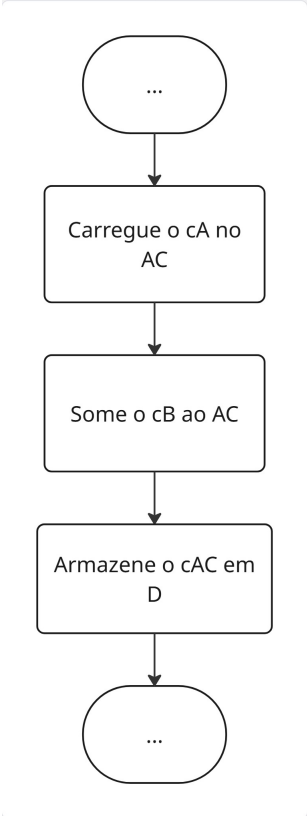


Figura 1.2: Algoritmo para o problema da soma de dois números. Fonte : Elaborado pelos autores.

Dessa forma, o procedimento efetua a soma dos valores dispostos na gaveta A e B e deixa disponível o resultado na gaveta D para uso futuro.

Procedimento: Retornar um valor ao usuário

Por fim, o procedimento de retornar o valor ao usuário é feito por meio da instrução "imprima cE"que faz com que Chico escreva o valor da gaveta E na folha de saída.

Programa : Juntando os procedimentos

Dessa forma, é possível juntar os procedimentos em sequência de modo a formar o programa que resolve o problema da soma de dois números inteiros. Considere a Figura 1.3 que representa a implementação no Computador HV-1, onde A, B e D representam números de gavetas quaisquer para os dois números inteiros e o valor da soma, respectivamente.

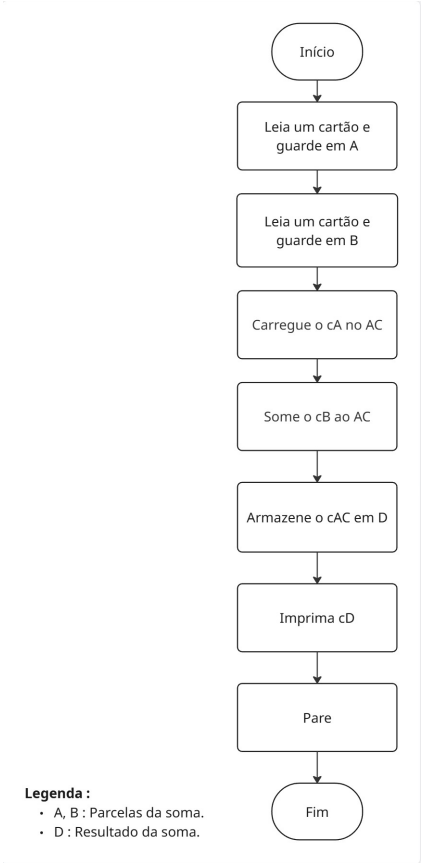


Figura 1.3: Programa para somar dois números inteiros no HV-1. Fonte : Elaborado pelos autores.

Observando o programa para resolver a soma de dois números é possível perceber que, no contexto do Computador HV-1, o programa consiste em uma sequência ordenada de cartões que definem um processo executado de maneira estrita por Chico.

Durante a execução do programa, se Chico se deparar com a instrução "leia um cartão e guarde em E", a execução é pausada até serem disponibilizados cartões com dados para que a instrução seja efetuada. Além disso, Chico continua em modo de execução até que ele encontre o cartão com a instrução "PARE", que denota o fim do programa.

Assim, é possível implementar o algoritmo para resolver o problema da soma de dois números utilizando o Computador HV-1.

1.2.3 Somando uma sequência de N números ($N > 2$)

Problema : *É dado um número $N > 2$ e uma sequência de N números inteiros. Determine a soma da sequência dos números utilizando o Computador HV-1 e retorne ao usuário.*

Problema : Somando N números

O problema em questão define uma operação de somatório com a quantidade de números N sendo um valor natural maior que dois e cada número da sequência um número inteiro, ou seja, negativo, nulo ou positivo de até três algarismos. É imposto pelo problema que o somatório esteja disponível para o usuário após o término da operação.

É importante ressaltar que fica implícito que a sequência de números é finita e perfeitamente calculável pela plataforma utilizada para implementação da resolução do problema.

Por fim, o problema pode ser formalizado matematicamente como o cálculo de T, tal que

$$T = \sum_{i=1}^{N} a_i$$

e onde a_i é um número inteiro e N um natural maior que 2. Vale destacar que a notação de somatório comprime simbolicamente múltiplas operações de soma.

Algoritmo : Somando N números

Para propor uma solução algorítmica ao problema do somatório é preciso, primeiro, dispor dos números envolvidos na soma, que são dados pelo usuário, efetuar uma operação de soma dos N números recebidos e retornar a soma ao usuário, uma estruturação possível a essa ideia é disposta na Figura 1.4

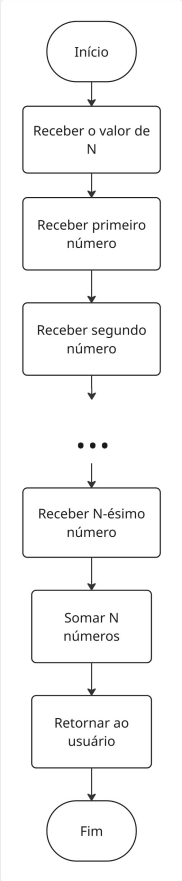


Figura 1.4: Algoritmo para o somatório de N números. Fonte : Elaborado pelos autores.

Contudo, perceba que a estrutura do algoritmo é dependente do valor de N, fato esse que é simbolicamente omitido pelo conceito do somatório e representado visualmente pela reticências no algoritmo apresentado. Assim, se N for igual a três, o algoritmo apresentará três passos de recebimento de número e uma soma de três números e se N for igual a 10, haverão dez passos de receber número e uma soma de dez números. Essa constatação impõe uma nova tarefa, nomeadamente, a de propor uma solução que independe do valor de N colocado pelo usuário.

Algoritmo : Soma de N números genérica

A chave para transformar o algoritmo que muda de estrutura à cada valor de N em um algoritmo genérico está no conceito de repetição. A repetição consiste em executar um conjunto fixo de

instruções sucessivas vezes enquanto uma condição é satisfeita, sendo esta condição expressa por meio de operadores lógicos e matemáticos.

Dessa forma, ao aplicar esse conceito no dado problema, tem-se que a instrução a ser repetida é a operação de receber um número e a condição é satisfeita enquanto a quantidade de números recebidos for menor que o valor de N inserido pelo usuário.

Perceba que, além do valor de N, torna-se necessário introduzir a quantidade de números inseridos pelo usuário para que seja utilizada a ideia de repetição aplicada à solução desse problema.

Vale notar também que a ideia de repetição é notável nos processos de construção de algoritmos, sendo uma das bases dos algoritmos e da programação, e define, no nível de programa, blocos de código chamados de estrutura de repetição. Tanto a ideia quanto o procedimento relacionado à repetição são reutilizáveis em uma ampla gama de problemas computacionais.

Como resultado da aplicação do conceito de repetição, é possível reescrever o algoritmo obtido anteriormente em um algoritmo genérico, isto é, que não muda sua estrutura com base nos valores inseridos, como o que está representado na Figura 1.5.

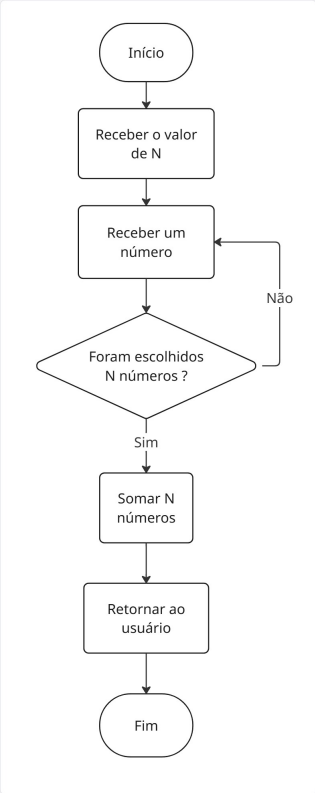


Figura 1.5: Algoritmo para o somatório genérico de N números. Fonte : Elaborado pelos autores.

É importante observar que o algoritmo possui um número fixo de instruções, mas define um programa que executa um número variável de passos a depender dos valores inseridos pelo usuário.

No entanto, este algoritmo ainda apresenta um pequeno problema, visível apenas se for adiantada a etapa de programa. O problema consiste na suposição de que a plataforma utilizada para implementar o programa definido pelo algoritmo, seja capaz somar N números concomitantemente ou apresentar alguma instrução para fazer soma com várias parcelas.

Como o Computador HV-1 só consegue fazer somas com duas parcelas, apresenta-se a subtarefa de alterar o algoritmo apresentado de modo que o passo de somar N números seja dividido em passos que envolvam apenas a soma de dois números.

Sub-Tarefa : Substituindo o passo da soma de N números

Para resolver essa sub-tarefa, nomeadamente, a de transformar o passo da soma de N números em passos que envolvem apenas a soma de dois números, é interessante regressar a análise conceitual do problema. Foi apresentando inicialmente o problema com base na notação de somatório, que, denota, em si, uma operação.

Como a necessidade é dividir essa operação em sucessivas operações menores de soma, se faz benéfico redefinir o problema na notação seguinte que define o somatório com base na quantidade de parcelas envolvidas, sendo a_N o n-ésimo termo do somatório :

$$Somatorio(N) = \begin{cases} a_1, N = 1 \\ a_N + Somatorio(N-1), N > 1 \end{cases}$$

Note que tal notação, da maneira apresentada, redefine o problema do somatório a partir de uma definição recursiva que pode ser utilizada de modo a estruturar um algoritmo com sucessivas somas de dois números, mesmo que o Somatório(N-1) seja uma compactação simbólica de várias operações. Tal abordagem estabelece sucessivas somas parciais que definem um valor que equivale ao subtotal do somatório, ideia compatível com o conceito de repetição.

Essa ideia originada na nova definição do problema do somatório também define um procedimento notável na programação que é a variável acumuladora, ou seja, uma gaveta responsável por acumular valores incrementáveis que, no caso, serão os subtotais relacionados à operação de somatório.

Recorrendo aos conceitos de repetição e acumulação é possível criar um algoritmo genérico e implementável no Computador HV-1.

O algoritmo resultante, representado na Figura 1.6, estabelece um fluxo onde o usuário insere o valor de N e começa a escolher as parcelas do somatório, tal que, cada novo número inserido é acumulado em um subtotal e, quando todos os números são escolhidos, o subtotal é retornado ao usuário e o algoritmo termina sua execução.

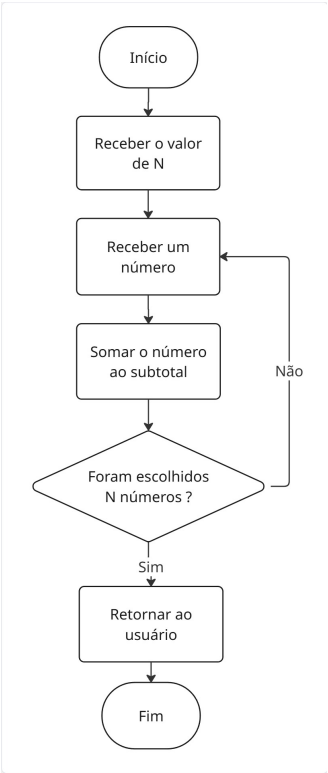


Figura 1.6: Algoritmo final para soma de N números. Fonte : Elaborado pelos autores.

Note que ao aplicar os conceitos de repetição e acumulação, transforma-se o algoritmo que apresentava uma estrutura interna dependente de um valor externo, em um algoritmo com estrutura fixa, que é capaz de somar uma sequência de N de números, com $N > 2$. Vale notar também que o subtotal e total são a mesma coisa durante a execução do programa, sendo a diferença dos dois puramente lógica : a quantidade de elementos adicionados.

Outra questão importante de se abordar quanto aos conceitos utilizados é que o conceito de repetição depende de outro conceito : a decisão. Perceba que para que a repetição continue é realizada uma pergunta : "Foram escolhidos N números ?"e sua resposta pauta a continuidade ou não do algoritmo. Esse conceito também é notável em algoritmos e programação e chama-se condição ou decisão.

As condições estabelecem fluxos alternativos que são seguidos na execução do algoritmo com base em um estado interno e escolhidos por operações lógicas. As operações envolvidas nas decisões são, no contexto do HV-1, majoritariamente, operação lógicas do tipo relacional, ou seja, operações de $>$ (maior que), $<$ (menor que), $=$ (igual a) e $\neq$ (diferente).

Feitas essas alterações tem-se um algoritmo implementável no Computador HV-1.

Programa : Somando N números

O algoritmo obtido para soma de uma sequência de N números envolve três conceitos novos, sendo eles a Acumulação, Decisão e Repetição. Tais conceitos apresentam certa complexidade inerente, tanto no nível lógico quanto no nível de implementação. Dessa forma serão dedicadas subseções separadas para explicação de como ocorre a implementação desses conceitos no Computador HV-1.

Procedimento : Acumulação

A implementação das variáveis de acumulação no computador HV-1 é mais próxima de um problema lógico do que sintático, pois não existe um comando específico para criá-las. Assim, definir uma variável de acumulação consiste em estabelecer, na lógica do programa, uma gaveta, ou seja, uma posição de memória, que exercerá essa função durante a execução do programa.

Procedimento : Estrutura de Decisão

Em relação à implementação das estruturas de decisão, há alguns pontos importantes a destacar. As decisões são tomadas com base em condições, como, por exemplo, verificar se a quantidade de números somados é igual a N. Dessa forma, a implementação dessas estruturas depende do comando de desvio condicional, que, no caso do Computador HV-1, redireciona a execução para outra instrução sempre que o valor do acumulador é diferente de zero.

Tendo em vista que esta é a única instrução disponível no HV-1 capaz de criar desvios de fluxo no programa, se faz necessário sempre criar condições que sejam denotadas por um valor diferente de zero no acumulador, deixando de ser intuitivo como implementar os operadores lógicos na plataforma.

Abstraindo a questão da condição é possível estabelecer o seguinte formato genérico de uma estrutura de decisão no computador HV-1, disponível na figura 1.7.

Dessa forma são criados dois fluxos que o programa pode seguir, um caso a condição seja satisfeita e outro caso a condição não seja satisfeita.

Procedimento : Estruturas de Repetição no HV-1

A diferença entre as estruturas de decisão e as estruturas de repetição reside no redirecionamento do fluxo de execução, controlado pelas condições.

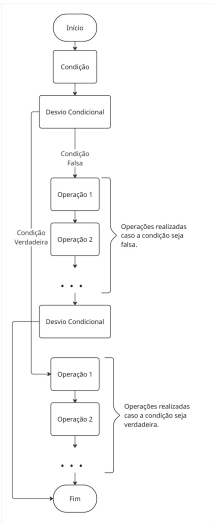


Figura 1.7: Fluxograma de uma estrutura de decisão genérica no HV-1. Fonte : Elaborado pelos autores.

Enquanto as estruturas de decisão criam ramificações alternativas de execução, as estruturas de repetição redirecionam o fluxo de volta ao mesmo ponto do programa, executando novamente as instruções até que a condição deixe de ser verdadeira.

Abstraindo novamente a questão das condições, é possível definir uma estrutura de repetição genérica como a que está representada na figura 1.8 a seguir :

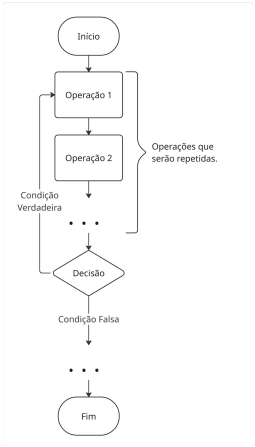


Figura 1.8: Fluxograma de uma estrutura de repetição genérica no HV-1. Fonte : Elaborado pelos autores.

Perceba que o passo decisão denota a construção de uma estrutura de decisão no qual o desvio condicional tem como alvo a primeira operação da repetição.

Programa : Juntando Procedimentos

Explicadas as implementações dos conceitos de acumulação, decisão e repetição é possível elaborar um programa para lidar com o problema do somatório.

O algoritmo obtido na seção anterior apresenta a leitura da quantidade de números a serem lidos e uma estrutura de repetição com as instruções receber um número e somar ao subtotal.

Uma questão relacionada é elaborar a condição que vai controlar a quantidade de repetições que vão ser executadas, que, no caso, é a quantidade de números que foram lidos.

Dessa forma, a quantidade de números lidos tem que ser igual ao número N, o que, em outras palavras, significa que N menos a quantidade de números lidos é diferente de zero enquanto não houverem sido lidos todos os N números.

Para resolver a contagem da quantidade de números lidos é possível adaptar o conceito de acumulação para contar o número de repetições e não agregar somas parciais. Dessa forma o código contará com uma estrutura de repetição no qual a condição que mantém o ciclo é que N menos a quantidade de números lidos é diferente de zero, onde a quantidade de números lidos é uma variável acumuladora que existe além do subtotal do somatório

A seguir se dispõe uma solução possível para o problema de somar uma sequência de N números, utilizando o computador HV-1 e os conceitos explanados anteriormente :

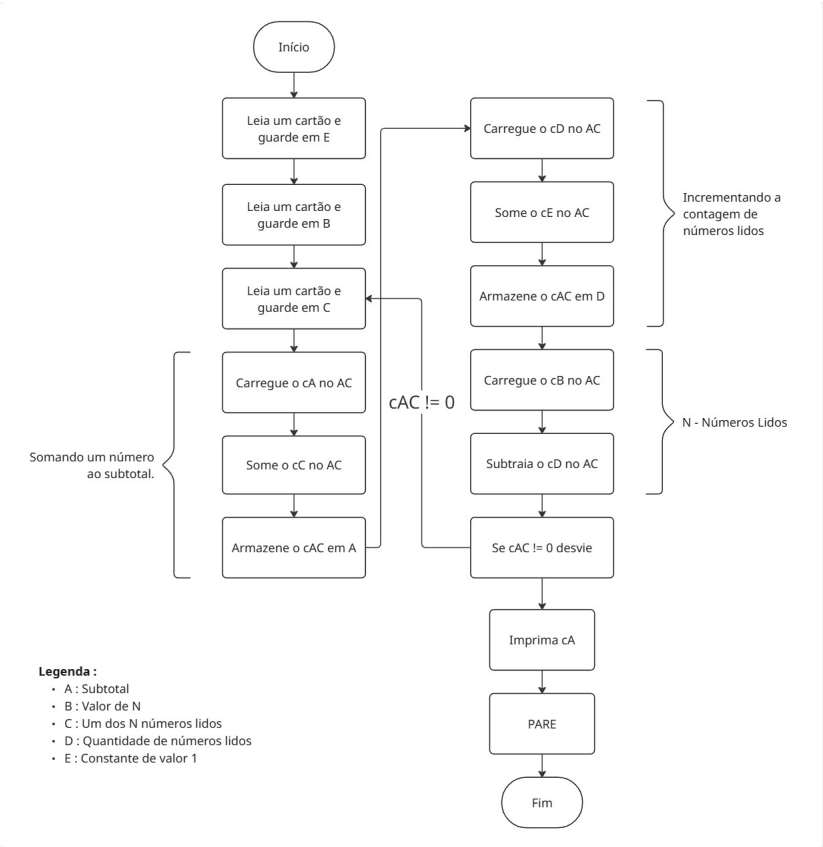


Figura 1.9: Resolução do problema de somar uma sequência de N números no computador HV-1. Fonte : Elaborado pelos autores.

Algumas observações são necessárias : o código contém cinco variáveis armazenadas nas gavetas denotadas por A, B, C, D e E e são respectivamente o subtotal, N, um número lido em uma repetição qualquer, a contagem de quantos números foram lidos e uma constate com valor um.

A presença da constate E é necessária pois o computador HV-1 não apresenta outros mecanismos para somar um ao valor de uma variável. Dessa forma ao inserir os cartões com os valores necessários para utilizar o código, o primeiro deles deve ser um cartão com o valor 1 para estabelecer a constante de incremento.

1.2.4 Somando uma sequência de números qualquer

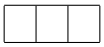
Desafio : *É dada uma sequência de números inteiros delimitadas por cartões com valor 000. Determine sua soma utilizando o HV-1.*

Esta seção é uma proposta de desafio para exercitar os conceitos apresentados anteriormente. Dessa maneira uma sugestão natural é utilizar os conceitos de estrutura de repetição e variável acumuladora para resolver esse problema.

Outra sugestão possível é de criar fluxogramas como os que foram apresentados ao longo do capítulo para entender como o algoritmo está funcionando.

1.3 O Computador HV-2

As instruções do computador HV-1 estão escritas por extenso, diferindo assim dos números armazenados em outras gavetas, como as de endereço 11 e 12 no programa exemplo da Sub-Seção 1.1.9. Conseguiremos uma grande simplificação de notação e de funcionamento se codificarmos as instruções, transformando-as também em número. Como veremos mais tarde, isso permitirá inclusive a substituição, com relativa facilidade, do operador CHICO por dispositivos eletrônicos. Para simplificar a compreensão, suponhamos que cada gaveta do gaveteiro contenha um quadro-negro da seguinte forma:



onde o CHICO só pode escrever números de 3 algarismos, como 001, 015, 152, etc. O novo computador assim obtido receberá a sigla HV-2.

No computador HV-2,as instruções deverão ser necessariamente codificadas como números de 3 algarismos, para podermos gravá-las no gaveteiro. Elas terão a forma CEE, onde C é um dígito de 1 a 7 e corresponde ao código da instrução; EE é um número de 00 a 99 e corresponde ao endereço da gaveta empregada na execução da instrução, denominado código de endereço. As instruções vistas na Seção 1.1.8 serão codificadas conforme a tabela dada a seguir:

instrução codificada	instrução
1EE	carregue o cEE no AC
2EE	armazene o cAC em EE
3EE	leia um cartão e guarde em EE
4EE	imprima o cEE
5EE	some o cEE ao AC
6EE	subtraia o cEE de AC
7EE	se cAC≠0 desvie para EE
8EE	pare

Lembremos que cEE significa conteúdo (agora sempre com 3 dígitos) da gaveta de endereço EE. Na instrução "pare" usamos sempre EE=00. Por exemplo, a instrução 512 encontrada pelo CHICO em alguma gaveta é interpretada por ele como "some o conteúdo da gaveta 12 ao conteúdo do acumulador e guarde o resultado no acumulador".

Observe um fato muito importante: é impossível, no modelo HV-2, distinguir-se o conteúdo de uma gaveta como correspondendo a uma instrução codificada ou a um número manipulado por certas instruções, o que não era o caso do modelo HV-1. Por exemplo, seguindo a execução do programa exemplo da Sub-Seção 1.1.9, vemos, na décima quarta linha, que a gaveta 11 recebe o conteúdo 105, correspondendo ao número "cento e cinco" (resultado da soma até esse momento) e não à instrução "carregue no AC o c05". Como pode o CHICO distinguir esses dois significados? Na verdade, a distinção é feita através da situação em que o CHICO se encontra ao se utilizar de uma gaveta (no caso, a 11). Assim, se ele estiver abrindo uma gaveta (no caso, a 11) à procura da próxima instrução a ser executada, o seu conteúdo será interpretado como sendo uma instrução codificada (no caso, a instrução 105). Por outro lado, se essa gaveta for aberta durante a execução de uma instrução, o seu conteúdo será usado como um valor numérico (no caso, o número 105).

A idéia de se armazenar as instruções da mesma maneira que os dados é atribuída ao famoso matemático americano John Von Neumann, que em meados da década de 1940 propôs esse esquema. Esse conceito foi um dos motivos que possibilitou a rápida evolução dos computadores daí para frente.

A codificação, por meio de números, de instruções que manipulam números é,em essência, a idéia fundamental. Uma idéia análoga foi aplicada na década de 1930 pelo matemático alemão Godel, o qual codificou numericamente teoremas sobre números, permitindo assim se enunciar teoremas sobre teoremas, chegando ao seu famoso "teorema da incompletude" dos sistemas axiomáticos.

1.4 Estudo de Caso no Computador HV-2

Como foi feito durante a seção anterior, será apresentado um estudo de como utilizar o Computador HV-2 para resolução de problemas computacionais. Mesmo que seja um modelo de computador diferente, o método desenvolvido no estudo do capítulo vai se manter, ou seja, as diferentes etapas de resolução serão mantidas e conceitos de algoritmos e programação serão reutilizados.

No entanto, uma questão que merece atenção e que foi citada anteriormente é o formato dos comandos do Computador HV-2, que, por serem totalmente numéricos, influenciam o processo de resolução dos problemas computacionais de duas formas : Há uma maior diferenciação da etapa de programação e código e pode causar confusão pois dados e comandos não apresentam diferença na forma.

A seguir são desenvolvidos os processos de resolução envolvendo o computador HV-2.

1.4.1 Média aritmética com três números

Problema : *São dados três números naturais. Calcule a média aritmética desses três números naturais e retorne para o usuário.*

Problema : Média aritmética com três números

A média aritmética é uma operação feita sobre uma quantidade de números e que resulta em um número que representa um valor médio dos números selecionados, onde cada valor utilizado para essa operação apresenta o mesmo peso ou "importância".

No contexto escolhido para resolução do problema, o Computador HV-2, deve-se considerar as limitações de trabalhar apenas com números inteiros, ou seja, tanto os números envolvidos na operação de média quanto o resultado serão números inteiros e que as operações matemáticas nativamente disponíveis são exclusivamente de soma e de subtração.

Independente do contexto, a operação de média aritmética é feita em dois passos : efetuar o somatório dos números envolvidos na operação e dividir esse somatório pela quantidade de números envolvidos na operação. No caso específico do problema apresentado, será necessário somar três números e dividir o resultado da soma por três.

Discutido o problema situado ao contexto do HV-2, é possível seguir para discussão de como resolver o problema citado.

Algoritmo : Média aritmética com três números

Observando o problema são necessários três tarefas : Coletar os três números inteiros, utilizá-los para fazer uma operação de média aritmética e disponibilizar o resultado ao usuário.

Para resolver esse problema é possível encarar a média como tarefa principal e a coleta dos números e disponibilização dos resultados como sub-tarefas que vão se adaptar ao modo que for escolhido para resolver a tarefa principal.

Como analisado na seção do problema, a média envolve duas operações menores : Somatório de uma sequência de números e uma operação de divisão. A primeira das operações foi analisada de forma detalhada no estudo de caso do HV-1 e seus resultados podem ser reutilizados no Computador HV-2, desde que sejam acompanhados das mudanças de sintaxe.

Dessa forma, o somatório será considerado como uma parte previamente feita da resolução do problema. Assim, basta utilizar o somatório dos números para efetuar a divisão por três e retornar ao usuário, como esboçado na Figura 1.10.

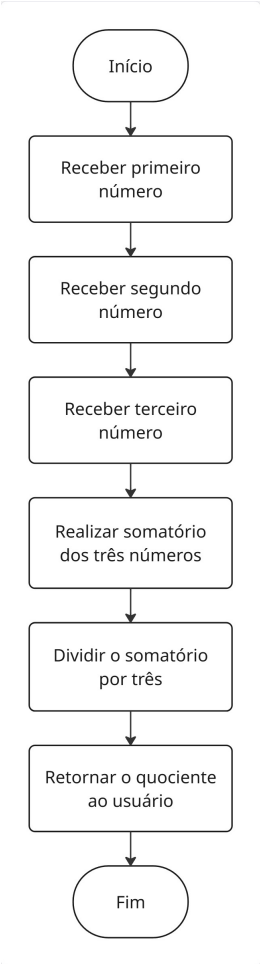


Figura 1.10: Algoritmo de média aritmética com três números. Fonte : Elaborado pelos autores

O algoritmo disposto na Figura 1.10 poderia ser suficiente para guiar uma implementação do problema de média aritmética com quantidade conhecida de números, contudo, ao adiantar a fase de análise do programa e implementação para o contexto dos computadores hipotéticos apresentados, é possível perceber que o Computador HV-2 não apresenta comando de divisão nativa, o que põe a necessidade de criar um procedimento para efetuar a operação de divisão.

Sub-Tarefa : Operação de divisão

Assim, será necessário estabelecer uma forma de efetuar operações de divisão de forma manual no Computador HV-2. De maneira simples, a operação de divisão pode ser definida como sucessivas operações de subtração, ou seja, uma repetição, feitas no dividendo pelo divisor, de modo que a condição de parada seja o dividendo acumulado ser menor que o divisor. Ao fim dessa operação, o quociente será equivalente à quantidade de subtrações realizadas e o resto o valor o dividendo quando este for menor que o divisor.

Reutilizando as estruturas de repetição definidas na seção do Computador HV-2, é possível efetuar a operação de divisão da forma disposta na Figura 1.11.

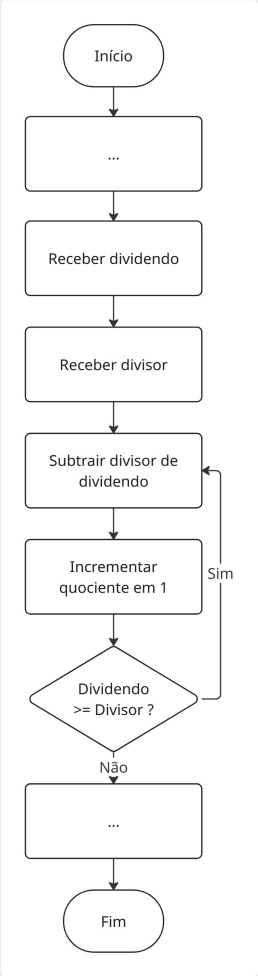


Figura 1.11: Algoritmo de operação de divisão. Fonte : Elaborado pelos autores

Contudo, note agora que o algoritmo envolve a operação relacional >=, que também não é uma operação nativa do Computador HV-2, ou seja, define um algoritmo não implementável no contexto do HV-2, o que cria mais uma sub-tarefa adicional, nomeadamente, a de estabelecer um processo manual para fazer a operação relaciona de maior ou igual.

Sub-Tarefa : Operador Maior ou Igual (>=)

Dessa maneira, se faz necessário criar uma maneira de efetuar a operação lógica de >= seguindo as limitações de instruções do Computador HV-2, ou seja, utilizando as operações de soma, subtração e ! = (diferença).

Uma maneira de emular o operador >= é primeiramente verificar a igualdade por meio de uma subtração, onde, se o resultado for igual a zero, os números são iguais.

Contudo, surge um problema ao avaliar os números por meio dos operadores de $> e <$ pois, no âmbito do Computador HV-2, só há operação para comparar se um valor é diferente de zero.

Assim, uma ideia possível depende do conceito de repetição : Enquanto os dois valores comparados forem diferentes de zero, decremente os valores em um, até que um deles fique igual a zero.

Supondo que A e B estavam sendo avaliados para ver se $A > B$, a condição só é válida se B atingir o valor zero antes de A.

Transformando essa ideia em um algoritmo, é possível obter algo parecido com o algoritmo representando na Figura 1.12.

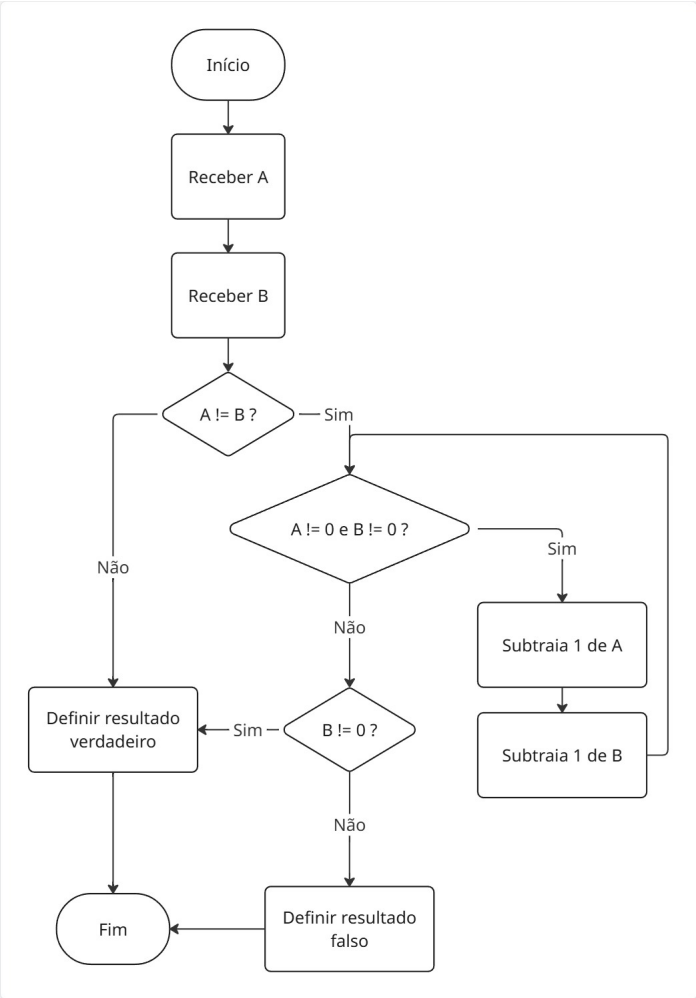


Figura 1.12: Algoritmo da operação relacional '>=' (Maior ou igual) usando como referência $A > B$. Fonte : Elaborado pelos autores

Note que esse algoritmo só funciona pois o problema estabelece que apenas números naturais estão envolvidos, pois, se houvessem números negativos, a operação de subtração nunca iria culminar no número zero e as instruções se repetiriam infinitamente ou o resultado ficaria distorcido pois apenas o número positivo poderia se igualar a zero por meio da sbutração.

Algoritmo : Média aritmética adaptada

Dessa forma, é possível aprimorar o algoritmo estabelecido inicialmente para atender as necessidades do ambiente de implementação, no caso o Computador HV-2, como foi feito na Figura 1.13.

Na criação do algoritmo foram utilizadas as abreviações S para Somatório e D para divisor, que, no caso, começa com o valor três. Tanto S quanto D envolvidos na operação relacional de maior ou igual devem ser cópias dos valores utilizados no restante do algoritmo por conta da necessidade de subtrair progressivamente os valores até zero para efetuar a comparação.

Perceba também que o quociente da operação de divisão é introduzida no meio do algoritmo, contudo, à depender do ambiente de implementação, é necessário declarar e, ou, inicializar um espaço dedicado ao valor do quociente. No caso do Computador HV-2, é necessário apenas deixar implícito na lógica do programa uma gaveta que vai conter o valor do quociente.

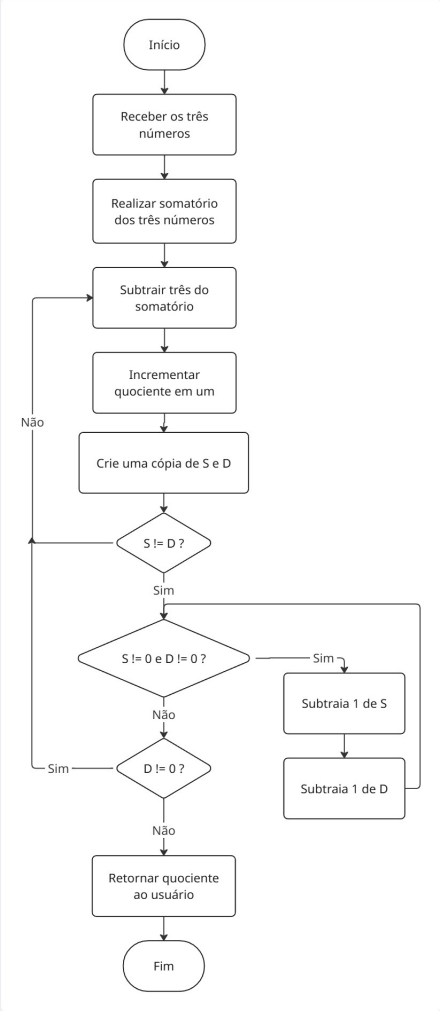


Figura 1.13: Algoritmo adaptado ao HV-2 de média aritmética com três números. Fonte : Elaborado pelos autores

1.4.2 Média aritmética ponderada com N números

Desafio : *É dada uma sequência de N números e uma sequência de N pesos em que cada número corresponde a um único peso, em ordem. Calcule a média aritmética ponderada dessa sequência de números e retorne ao usuário .*

1.5 O Computador HIPO

O fabricante dos computadores HV-2 percebeu, em um dado instante, que eles tinham as seguintes desvantagens:

1. os valores numéricos que podiam ser representados nas gavetas eram muito restritos, permitindo apenas 3 algarismos, sem sinal;

2. o operador CHICO era muito lento;

3. da mesma maneira, o mecanismo das gavetas e de outros componentes também era muito lento. Assim, propôs-se a fabricar um outro computador, que denominou de HIPO (cuja sigla provém de "computador hipotético"). Podemos descrever a sua organização comparando-a com a do HV-2.

1.5.1 Memória

Em lugar do gaveteiro, o HIPO dispõe de um dispositivo eletrônico, denominado **memória**, com regras de funcionamento análogas às do gaveteiro do HV-2. Este dispositivo contém partes, denominadas **células**, que correspondem às gavetas no gaveteiro. Cada célula tem comprimento de 8 bits (1 byte). O modelo mais simples do HIPO é produzido com memória de 32 células, de endereços 00 a 31.

Em cada célula podem ser representadas instruções codificadas como especificado mais adiante ou um número inteiro de -128 a 127, representado por complemento de 2 (dois).

1.5.2 CPU

No caso do computador HV-2, um operador, o CHICO, acionava todos os dispositivos, seja gaveteiro ou calculadora. No HIPO, esse papel é desempenhado por um sistema de circuitos eletrônicos, cujo funcionamento equivale às ações executadas pelo CHICO no HV-2. Esse sistema é denominado de **Unidade de Controle** ou simplesmente UC.

Na seção 1.1.6 foi dito que o CHICO pode estar em um dos dois estados: "carga" e "execução". Analogamente, a unidade de controle do HIPO estará um um desses dois estados em qualquer instante. O CHICO interpretava as instruções escritas nas gavetas do HV-2. As instruções interpretadas por essa UC do HIPO e armazenadas na memória têm o formato mostrado a seguir:

C	C	C	E	E	E	E	E

onde os dígitos binários "CCC" representam o código da instrução e os dígitos binários "EEEEE" representam o endereço de uma célula de memória.

No lugar da calculadora, o computador HIPO realiza as operações aritméticas por meio de um conjunto de circuitos denominado **Unidade Lógica e Aritmética** ou simplesmente ULA. Para executar uma operação de soma, por exemplo, impulsos eletrônicos são encaminhados à seção apropriada do circuito da ALU, iniciando uma sequência de operações que resulta na obtenção do resultado da operação no acumulador. O acumulador é um registrador acessível eletronicamente, isto é, não possui exibição visual.

1.5.3 EPI

O endereço da próxima instrução é um registrador eletrônico, sem exibição visual, com formato de um número com 5 algarismos binários. Somente a CPU do HIPO tem acesso ao EPI, podendo consultá-lo ou alterar seu conteúdo.

1.5.4 Unidade de Entrada

Em lugar do porta-cartões, o HIPO contém uma unidade de entrada com capacidade para ler eletronicamente **linhas de entrada** com o formato apresentado a seguir:

I/D :□□□□□□□□ e E :□□□□□.

Isto é, cada linha de entrada contém duas partes. Se a unidade de controle está em estado de execução, só é utilizada a parte I/D (iniciais de Instrução/Dado). O usuário especifica na mesma um número de 8 dígitos binários, onde o dígito mais significativo representa o sinal do número (0-não negativo e 1-negativo). Uma instrução de leitura executada pela unidade de controle provoca a transcrição do número dado pelo usuário na linha de entrada para uma célula da memória. O endereço dessa célula é especificado pela instrução de leitura.

No estado de carga, o usuário especifica em I/D uma instrução conforme o formato dado na Seção 1.5.2 e, em E, o endereço da célula de memória onde a instrução deve ser carregada. O

mesmo esquema do HV-1 é usado para se mudar a unidade de controle do estado de carga para o estado de execução e vice-versa.

Vários dispositivos podem ser usados como unidade de entrada: cartões perfurados (onde furos codificam os elementos das linhas), cartões marcados a lápis ou com tinta magnética, teclado como de máquina de escrever, etc. Todos eles transformam a representação externa acessível ao usuário em impulsos eletrônicos que são enviados pela unidade de controle à memória, posicionando os circuitos desta a fim de que as células envolvidas tenham um conteúdo equivalente à representação externa.

1.5.5 Unidade de Saída

Em lugar da folha de saída do HV-2, o HIPO contém uma unidade de saída com capacidade de gravar **linhas de saída**. Cada uma destas consiste em um número com 8 dígitos binários, onde o dígito mais significativo indica o sinal do número (0-não negativo e 1-negativo).

Novamente, nenhum dispositivo de saída particular foi indicado, podendo o mesmo ser uma máquina de escrever, uma impressora de linha, um terminal de vídeo, etc.

1.5.6 Instruções do HIPO

Vejamos algumas das instruções do computador HIPO. Supomos que as instruções tenham código de endereço "EEEEE". Lembramos que cAC abrevia "conteúdo do acumulador".

Instrução Codificada	**Significado**
001EEEEE	carregue o cEE no AC
010EEEEE	armazene o cAC em EE
011EEEEE	leia um cartão e guarde em EE
100EEEEE	imprima o cEE
101EEEEE	some o cEE ao AC
110EEEEE	subtraia o cEE de AC
111EEEEE	se cAC≠0 desvie para EE
000EEEEE	inicia estado de execução na gaveta EEEEE
00000000	pare

Devido à complexidade adicionada pela digitalização, transição entre a execução gerenciada pelo operador CHICO e a execução gerenciada pela Unidade de Controle - UC, e a consequente adequação dos comandos ao sistema binário tornou-se benéfico a adoção de códigos mnemônicos para cada um dos comandos do computador HIPO.

Código Mnemônico	**Instrução Codificada**	**Significado**
CAR	001EEEEE	carregue o cEE no AC
ARM	010EEEEE	armazene o cAC em EE
LNC	011EEEEE	leia um cartão e guarde em EE
IMP	100EEEEE	imprima o cEE
SOM	101EEEEE	some o cEE ao AC
SUB	110EEEEE	subtraia o cEE de AC
DES	111EEEEE	se cAC≠0 desvie para EE
INI	000EEEEE	inicia estado de execução na gaveta EEEEE
PAR	00000000	pare

Dessa maneira a unidade de controle consegue processar comandos fornecidos pelo usuário à partir de seu correspondente código mnemônico acompanhado por um número no formato EEEEE indicando a célula da memória que vai ser manipulada pela instrução.

Exemplo . Para realizar a operação de ler um número e armazenar na célula de endereço 01000, o usuário deve utilizar o comando : *LNC 01000*.

1.6 Conclusão

O texto apresentado neste Capítulo foi retirado e adaptado, com fins didáticos, do Capítulo 2 "O Computador à Gaveta", do livro "Introdução à Computação e à Construção de Algoritmos"[5].

Referências Bibliográficas

[1] Michal Armoni and Liat Nakar. On teaching abstraction in computer science: Secondary-school teachers' perceptions vs. practices. *UKICER 23: Conference on United Kingdom Ireland Computing Education Research*, 2023.

[2] Michal Armoni and David Statter. Teaching abstraction in computer science to 7th grade students. *ACM Transactions on Computing Education (TOCE)*, 2020.

[3] Quintin Cutts, Sarah Esper, Marlena Fecho, Stephen R Foster, and Beth Simon. The abstraction transition taxonomy: Developing desired learning outcomes through the lens of situated cognition. In *Proceedings of the ninth annual international conference on International computing education research*, pages 63–70, 2012.

[4] Sue Sentance, Jane Waite, and Maria Kallia. Teaching computer programming with primm: a sociocultural perspective. *Computer Science Education*, 29(2-3):136–176, 2019.

[5] Routo Terada and Waldemar W. Setzer. *Introdução à Computação e à Construção de Algoritmos*. Makron Books do Brasil, 1992.