Calculadora em Assembly x86

Escola Superior de Tecnologia de Tomar,
Licenciatura de Engenharia Informática
Arquitetura de Computadores

Rúben B. M. Gomes Cardoso
Nº 23885

Rodrigo Miguel Barreto Serra
Nº 24180

Janeiro, 2022

Resumo

No âmbito da unidade curricular Arquitetura de Computadores, foi solicitado um algoritmo em Assembly x86 (desenvolvido utilizando o Emu8086) que fosse capaz de calcular Divisões e Raízes Quadradas, utilizando valores fornecidos pelo utilizador através de uma interface gráfica que permitisse a escolha entre ambos os algoritmos previamente mencionados. O utilizador teria ainda a opção de optar por utilizar o teclado ou selecionar botões na interface de forma a escolher os valores pretendidos.

Palavras-chave— Assembly x86, Divisão Inteira, Raiz Quadrada, Emu8086

Conteúdo

1 Interface Gráfica **3**
- 1.1 Objetivo . . . 3
- 1.2 Pseudocódigo . . . 3
  - 1.2.1 Tela inicial . . . 3
  - 1.2.2 Tela de Input . . . 3
- 1.3 Detalhes de funcionamento . . . 4
  - 1.3.1 Inserção de dígitos . . . 4
  - 1.3.2 Visualização do input . . . 4
  - 1.3.3 Backspace . . . 4
  - 1.3.4 Negativo . . . 4
- 1.4 Fluxograma . . . 5
  - 1.4.1 ShowMainScreen . . . 5
  - 1.4.2 WelcomeWindow . . . 5
  - 1.4.3 InsideSquareText . . . 5
  - 1.4.4 DesenhaQuadrados . . . 5
  - 1.4.5 RecebeMouseKeyboardInput . . . 6
  - 1.4.6 KbHandlerInputScreen . . . 6

2 Algoritmo da Divisão **7**
- 2.1 Objetivo . . . 7
- 2.2 Pseudocódigo . . . 7
- 2.3 Fluxograma . . . 8

3 Algoritmo da Raiz Quadrada **9**
- 3.1 Objetivo . . . 9
- 3.2 Pseudocódigo . . . 9
- 3.3 Fluxograma Geral . . . 10

4 Problemas no desenvolvimento **11**

Capítulo 1

Interface Gráfica

1.1 Objetivo

Para permitir a obtenção dos dados escolhidos pelo utilizador foi necessário utilizar uma interface gráfica capaz de o fazer. Esta interface deveria permitir a escolha entre os dois algoritmos utilizando ou o rato ou o teclado como meio de input. A interface iria ainda precisar de mostrar os resultados dos algoritmos após a sua execução.

1.2 Pseudocódigo

1.2.1 Tela inicial

1. Imprimir um texto introdutório que indique o propósito do programa
2. Desenhar os quadrados que representam botões onde o utilizador poderá clicar
3. Ler o input do teclado que permite ao utilizador escolher se pretende utilizar o teclado ou o rato tanto para introduzir os inputs, como para escolher qual algoritmo pretende executar

1.2.2 Tela de Input

1. Desenhar os 14 botões necessários para acomodar os inputs possíveis
2. Escrever o texto dentro dos botões todos os inputs possíveis[I]
3. Mostrar o texto "Dividendo"ou "Radicando", de forma a indicar ao utilizador que pode inserir o valor pretendido
4. Ler o teclado e adicionar os valores inseridos a um array
5. Mostrar o array do passo acima consoante o utilizador adiciona ou remove valores
6. Mostrar o texto "Divisor"(caso seja a tela de input da divisão), de forma a indicar ao utilizador que pode inserir o valor pretendido
7. Ler o teclado e o rato e adicionar os valores inseridos a um array
8. Mostrar o array do passo acima consoante o utilizador adiciona ou remove valores
9. Executar o respetivo algoritmo e mostrar o resultado juntamente com o texto "Resultado"

[I] Algarismos de 0-9, **'-'**, **','**, backspace, confirmação

1.3 Detalhes de funcionamento

1.3.1 Inserção de dígitos

Para a inserção de dígitos foi utilizado a interrupção 16H que permite a leitura do teclado e devolve o código ASCII do caractere pressionado. Sabendo o código da tecla pressionada basta verificar se este está dentro do intervalo aceite, neste caso, dígitos de 0-9 e caso se encontre dentro de cujo intervalo adiciona-se o algarismo[I] ao respetivo array.

Por motivos de compatibilidade com o algoritmo da divisão previamente desenvolvido foi necessário armazenar o divisor em formato de número e não só dentro de um array. Para desenvolver esta funcionalidade, foi criada uma nova variável que armazenaria o divisor, onde iriamos concatenar [II]os dígitos inseridos á medida que fossem selecionados.

1.3.2 Visualização do input

Para que o utilizador conseguisse visualizar os dados inseridos utilizou-se a interrupção 21H para imprimir os dígitos na janela da aplicação. Sempre que o utilizador inseria um digito no respetivo array, esse mesmo valor era escrito utilizando a interrupção 21H.

1.3.3 Backspace

No desenvolvimento da funcionalidade do Backspace foi utilizada a interrupção 21H novamente. Visto que impressão do caractere do backspace não remove o caractere anterior e ao invés disso, volta uma casa atrás, podemos utilizar esta funcionalidade para recuar um digito, imprimir um espaço[III] e imprimir novamente um backspace para recuarmos uma casa.

Porém, remover o ultimo caractere impresso não remove o mesmo do array onde estão a ser armazenados todos os dígitos escolhidos, para resolver esse problema basta decrementar a posição atual do array por 1. Desta forma, o próximo digito inserido irá substituir no array o caractere que foi eliminado.

Visto que ainda temos a variável que armazena o divisor em formato de número, é necessário dividir esta variável por 10 de forma a remover o digito mais á direita, ou seja, o ultimo digito adicionado.

1.3.4 Negativo

Esta funcionalidade é apenas utilizada no algoritmo da divisão

Visto que numa divisão é possível a utilização de valores negativos foi necessário implementar um método que permitisse simbolizar que os números eram negativos. A norma é colocar o sinal de negativo antes do número e para facilitar a implementação foi decidido que se deveria limitar o input para que apenas fosse possível inserir o '-'. antes de qualquer digito.

Para fazer esta verificação, basta confirmar que o tamanho do array do dividendo/divisor[IV] é igual a 0, querendo isto dizer que o array está vazio e que o '-' será o primeiro caractere.

É ainda necessário impedir a inserção de dois '-', para isto foi utilizada a flag implementada no algoritmo da divisão que indica se um número é negativo ou não. Considerando que o dividendo é o primeiro valor inserido, este só pode ter a flag a 1 ou 0, caso seja negativo ou não respetivamente. Logo se $flag = 1$ impedimos que o utilizador coloque outro '-', utilizando a mesma lógica para o divisor.

[I] Visto que os códigos ASCII dos dígitos 0-9 começam no 48 é necessário subtrair esse valor ao código original de forma a adicionar o digito correto ao array

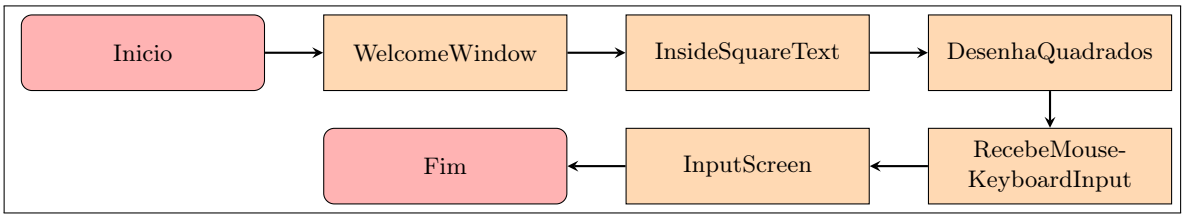
[II] Utilizando $var = var \times 10 + digito$

[III] De forma a remover o caractere onde se encontra o cursor

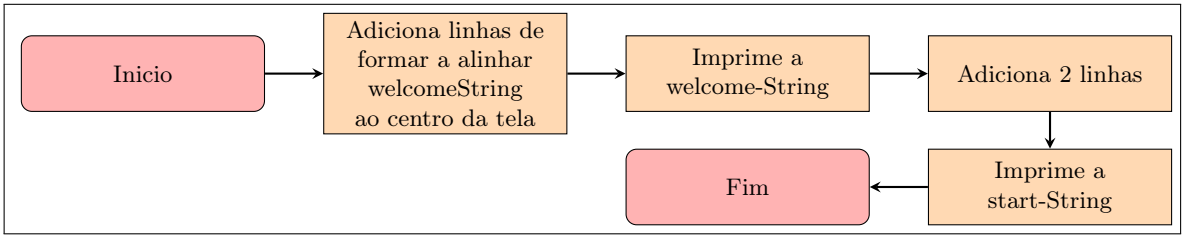
[IV] Dependendo de qual deles está a ser inserido

1.4 Fluxograma

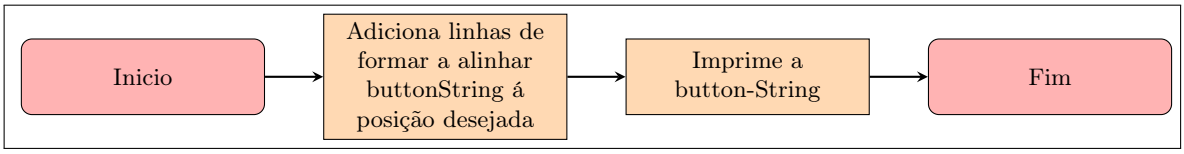
1.4.1 ShowMainScreen



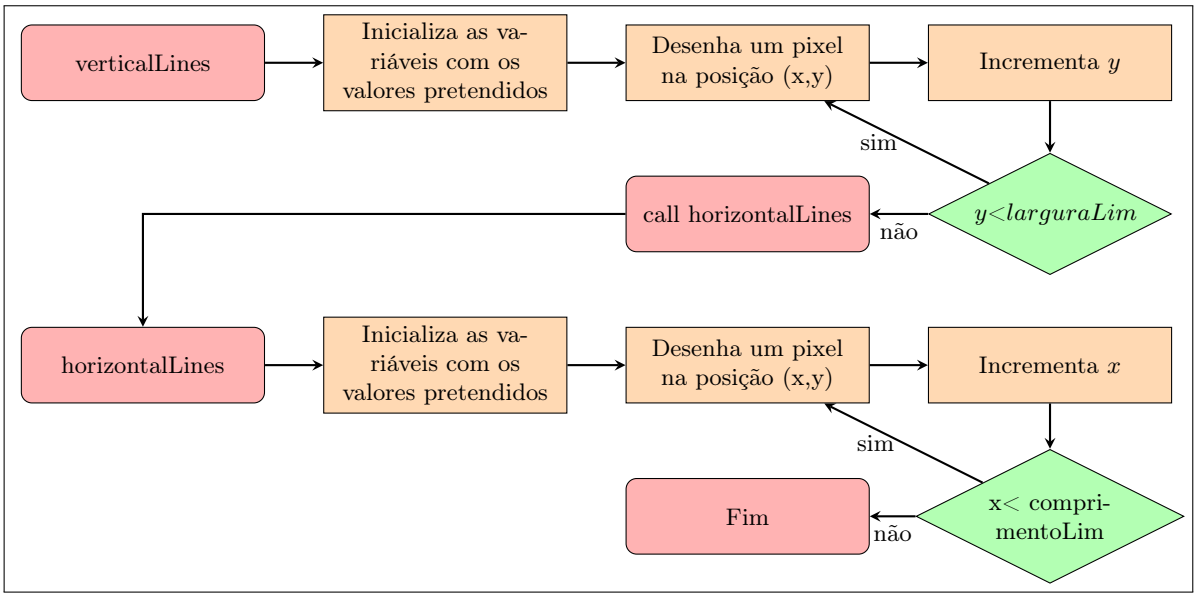
1.4.2 WelcomeWindow



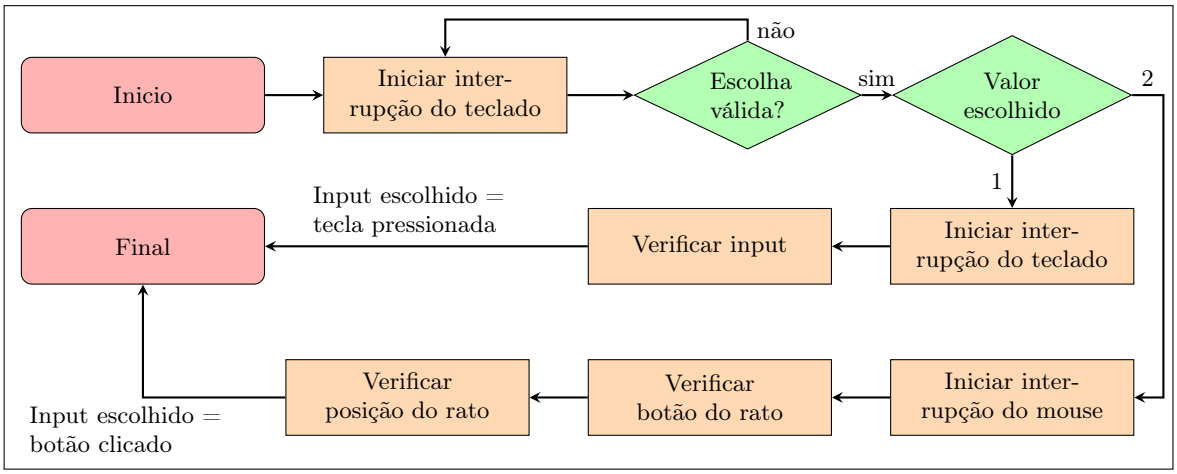
1.4.3 InsideSquareText



1.4.4 DesenhaQuadrados

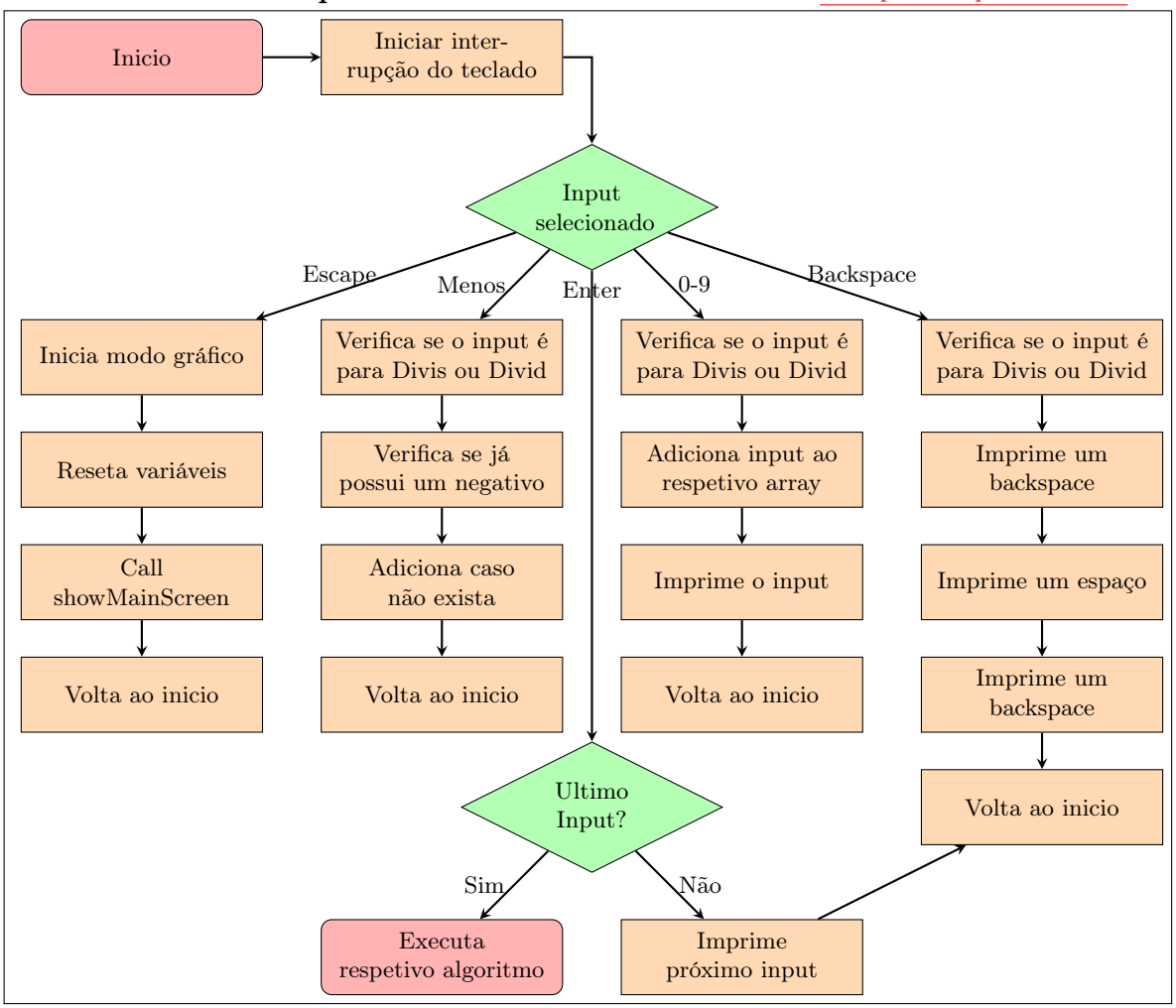


1.4.5 RecebeMouseKeyboardInput



1.4.6 KbHandlerInputScreen

Exemplo do input da divisão



Capítulo 2

Algoritmo da Divisão

2.1 Objetivo

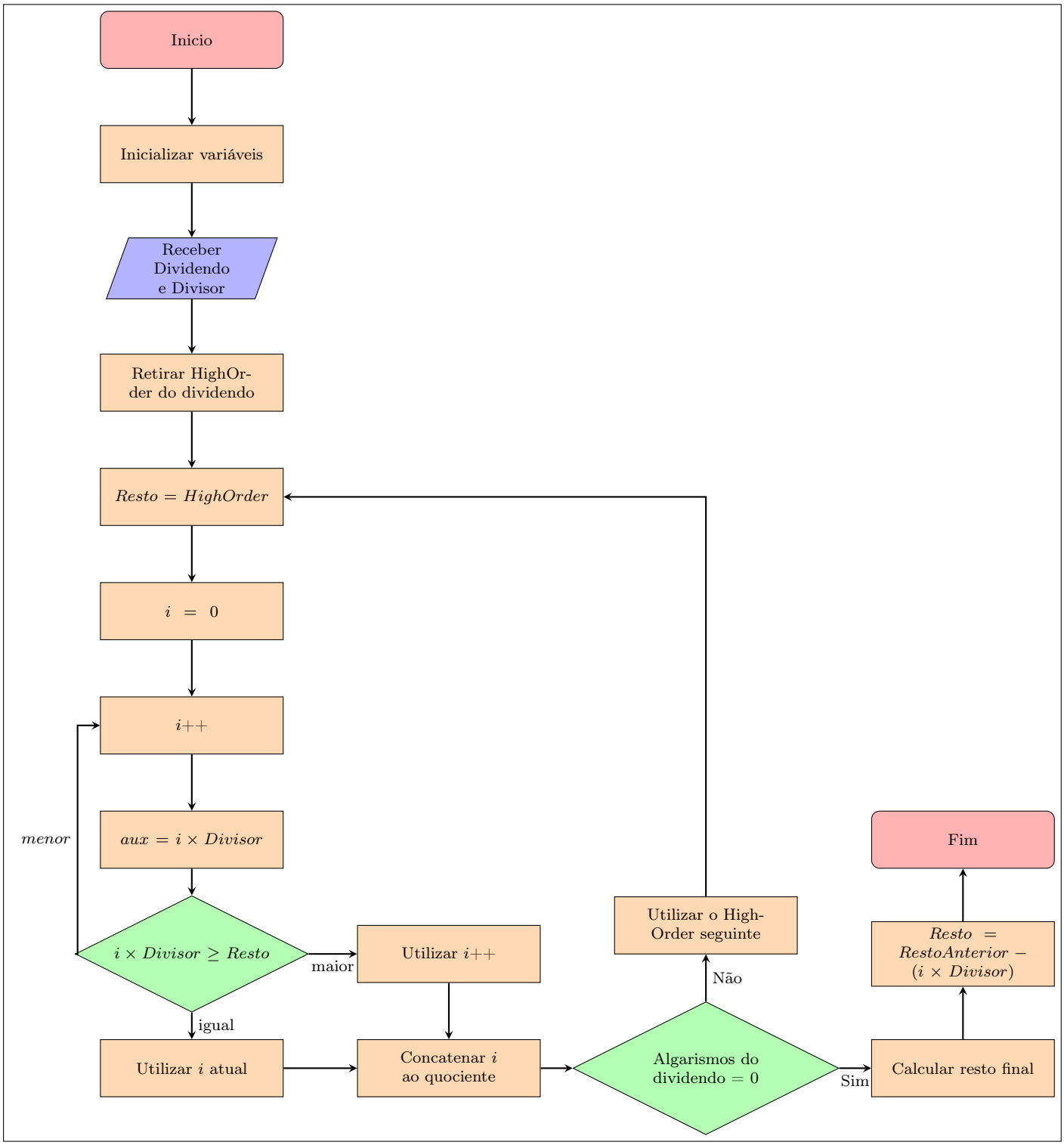
Este algoritmo pretende receber as variáveis, dividendo, e divisor (podendo estas ser negativas) e em seguida executar o algoritmo da divisão de forma a devolver o resultado da divisão, bem como, o resto da mesma.

2.2 Pseudocódigo

1. Inicialização das variáveis necessárias[I]
2. Retirar o primeiro HighOrder do dividendo e atribuir o seu valor á variável Resto
3. Iterar as vezes necessárias até a operação i[II] $\times$ $Divisor > Resto$
4. Após a condição ser satisfeita:
   (a) Utilizar o valor atual da variável i caso, $i \times Divisor = Resto$
   (b) Realizar o cálculo $i = i - 1$,caso, $i \times Divisor > Resto$
5. Concatenar o valor de i á variável Quociente
6. Verificar se existem mais algarismos no dividendo
   (a) Caso existam, voltar ao passo 3. com o novo valor retirado do dividendo
   (b) Caso não existam, obter o valor do resultado a partir da expressão, $Resto = Resto - (i \times Divisor)$

[I] Este pseudocódigo assume que as variáveis Divisor e Dividendo já foram obtidas através da interface gráfica
[II] A variável i é utilizada como variável de iteração em ciclos

2.3 Fluxograma



Capítulo 3

Algoritmo da Raiz Quadrada

3.1 Objetivo

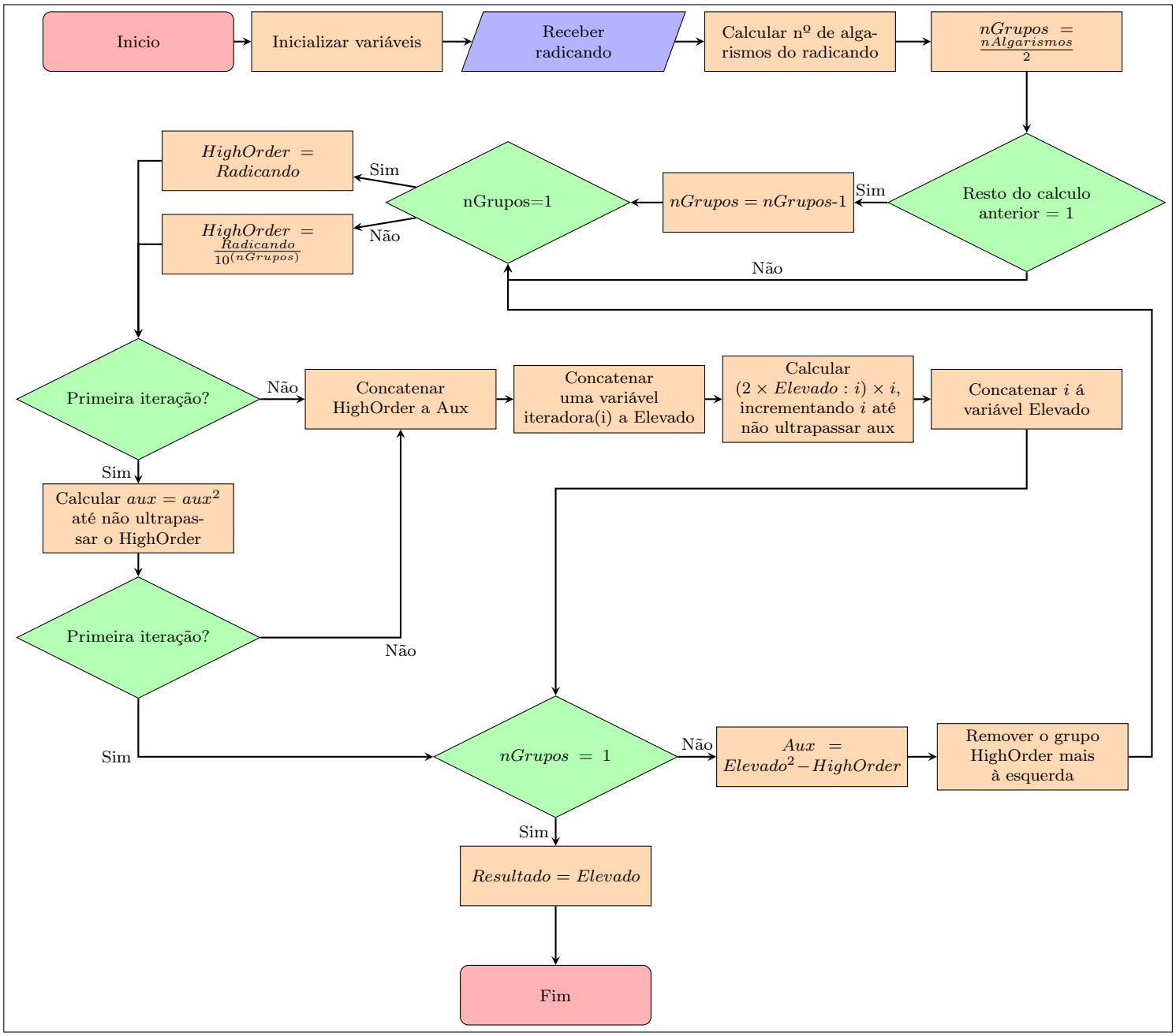
Este algoritmo tem como objetivo receber o valor do Radicando e em seguida calcular a raiz quadrada do valor inserido, e devolver o seu resultado.

3.2 Pseudocódigo

1. Inicializar as variáveis necessárias[I]
2. Utilizar divisões consecutivas por 10 de forma a obter o número de dígitos do Radicando e armazenar o resultado em nAlgarismos
3. Obter o número de pares possíveis através da formula, $nGrupos = \frac{nAlgarismos}{2}$
   (a) Caso o resto da divisão do passo anterior seja igual a 1, realizar $nGrupos = nGrupos + 1$
4. Verificar o valor de nGrupos
   (a) Caso existam múltiplos grupos, ou seja $nGrupos > 1$, obter o grupo de HighOrder através da fórmula, $HighOrder = \frac{Radicando}{10^{(nGrupos)}}$
   (b) Caso exista um único grupo atribuir o valor do Radicando ao HighOrder
5. Caso seja a primeira iteração, realizar $aux \times aux$[II] de forma a encontrar o maior número possível que não ultrapasse o HighOrder e armazenar a variável valor da variável aux na variável Elevado
6. Caso não seja a primeira iteração, concatenar o novo HighOrder á variável aux
   (a) Caso seja a primeira iteração saltar para o passo 9.
7. Calcular $(2 \times Elevado : a) \times a$, incrementado o valor de a enquanto o resultado não for superior a aux
8. Concatenar o valor de a obtido no passo anterior á variável Elevado
9. Parar a execução do algoritmo caso $nGrupos = 1$ e armazenar o valor de Elevado na variável Resultado
10. Calcular $Aux = HighOrder - Elevado^2$
11. Calcular $Radicando = Radicando - (HighOrder \times 10^{nGrupos})$, de forma a remover o grupo mais á esquerda.
12. Saltar para o passo 4.

[I] Este pseudocódigo assume que a variável Radicando já foi obtida através da interface gráfica
[II] Variável auxiliar inicializada a 0

3.3 Fluxograma Geral



Capítulo 4

Problemas no desenvolvimento

- Visto que os algoritmos da Divisão e Raiz previamente desenvolvidos não suportavam números decimais, foi escolhido não adicionar na tela de input o botão da ',.

  Isto poderia ser corrigido adicionando uma variável que contasse o número de dígitos antes do ponto decimal e alterando os cálculos para suportar esta funcionalidade.

- À custa da variável de Divisor ser armazenada não num array mas como um número, não é possível utilizar valores superiores a 65536, visto que qualquer valor superior ao supramencionado irá dar overflow no programa.

  A forma de resolver esta falha seria alterar o código de forma a armazenar a variável Divisor num array.

- Existem certos casos, nomeadamente no algoritmo da Divisão, onde a concatenação com 0 falha, visto que em números inteiros, os $0s$ à esquerda são ignorados.

  Uma possível forma de resolver este problema é armazenar as variáveis, onde pode ser necessário concatenar com um zero á esquerda, dentro de um array.

- Devido a um desentendimento sobre os objetivos do trabalho e uma compreensão tardia sobre o que realmente era solicitado, a tela de input funciona à base do teclado e não do cursor do mouse como terá sido pedido.

  Uma solução seria suportar a interrupção 33H, responsável pelo rato, em simultâneo com a interrupção 16H do teclado.

- Por falta de melhor conhecimento na altura do desenvolvimento, a utilização dos dois métodos de input (Mouse e Teclado) é possível na tela inicial, porém o utilizador tem primeiro que escolher o método desejado utilizando o teclado.

  Este problema seria evitado se fosse utilizada a interrupção do mouse enquanto não fosse detetado o clique de uma tecla

- Visto que os arrays onde são armazenados os dígitos necessários, são inicializados com um valor fixo de 10 casas, não é possível a utilização de números com mais de 10 algarismos.

  Uma possível solução, embora não ótima, seria inicializar o array com um número exagerado de casas de forma a permitir mais dígitos. Porém, esta solução não resolve o problema e seria mais correto simplesmente limitar o input a 10 casas.

Conclusão

Apesar de existirem alguns problemas no programa final, e da utilização de uma linguagem focada para uma arquitetura já bastante desatualizada, limitando assim as possibilidades dessa mesma linguagem, é possível afirmar que o programa se encontra num estado funcional utilizando uma grande porcentagem das funcionalidades solicitadas para o mesmo.