

UNIVERSIDADE POSITIVO
GABRIELA CRISTINA SCHMITT
LUCAS JULIANO POSSA GOMES

ARQUITETURA DE SISTEMAS COMPUTACIONAIS

CURITIBA
2022

GABRIELA CRISTINA SCHMITT
RGM: 25733150
LUCAS JULIANO POSSA GOMES
RGM: 25798472

ARQUITETURA DE SISTEMAS COMPUTACIONAIS
SIMULADOR DE MEMÓRIA CACHE

Trabalho de desenvolvimento de aplicação apresentado como requisito parcial para a disciplina de Arquitetura de Sistemas Computacionais no curso de Ciência da Computação da Universidade Positivo.

Orientador(a): Prof. Rodrigo Machniewicz Sokulski

CURITIBA
2022

SUMÁRIO

1	INTRODUÇÃO	4
2	IMPLEMENTAÇÃO	5
2.1	ESTRUTURA DE DADOS	5
2.2	FLUXOGRAMA	6
2.3	COMPILAÇÃO E AMOSTRA DE USO	7
3	CONCLUSÃO	8

1 INTRODUÇÃO

Este trabalho tem como objetivo desenvolver um simulador de memória cache L1, onde o processador realiza requisições de leitura e escrita de dados na memória. A entrada dos dados para as requisições é feita pelo caminho completo de um arquivo de texto (.in) informado na inicialização do programa, e a saída dos dados é executada via stdout.

Conforme o plano de orientações para o trabalho, a soma dos dígitos menos significativos do RGM dos alunos Gabriela e Lucas, resultou em 2, portanto a configuração a ser programada é uma cache L1 com associatividade 16, política de substituição FIFO, mecanismo de coerência de escritas Write-Back e política de alocação durante escritas Write-Allocate.

2 IMPLEMENTAÇÃO

Para entender melhor o procedimento necessário para o programa, foi feito uma pesquisa no GitHub de trabalhos correlatos onde foi escolhido o seguinte exemplo <https://github.com/GEEGABYTE1/Dolphin>.

Trata-se de um algoritmo em Python, simulador de memória cache L1 com associatividade 2, política de substituição FIFO, mecanismo de coerência de escritas Write-Back e política de alocação durante escritas Write-No-Allocate. Após o entendimento deste algoritmo, foi feito um algoritmo em Python para o presente trabalho, conforme as orientações citadas na introdução.

Depois de escrito o algoritmo, finalmente foi feito o código em C.

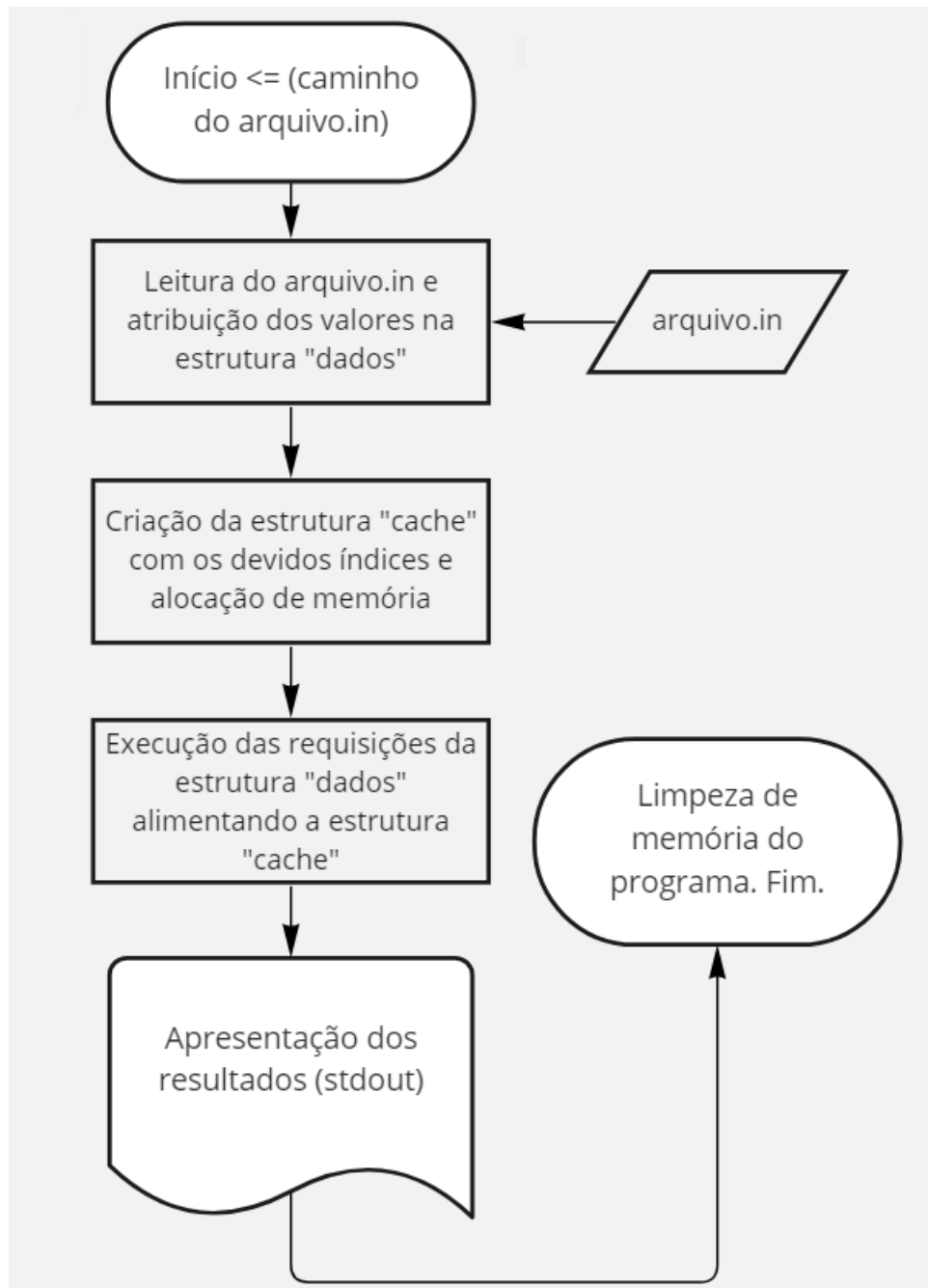
2.1 ESTRUTURA DE DADOS

A estrutura de dados em sua totalidade está declarada no arquivo “cache.h”. Foi feito uma estrutura simples chamada “cabeçalho” para armazenar as configurações da cache e dos arquivos de entrada e outra estrutura chamada “requisição” onde para cada linha de requisição do arquivo de entrada, é instanciada uma estrutura deste tipo que armazena os dados de uma requisição. Por fim, uma estrutura composta destas duas, chamada “dados” irá armazenar um cabeçalho, e N requisições, bem como as variáveis “miss”, “hit” e “tempo total” que são acrescentadas ao longo da execução do programa.

Para estruturar a memória cache, foi declarada uma estrutura chamada “bloco”, que armazena as informações de um bloco de cache, onde a mesma possui os campos “tag”, “validade” e “dado”, também foi acrescentado um campo chamado “clock” para facilitar o algoritmo da política de substituição. Depois disso, uma segunda estrutura composta, chamada “cache” foi declarada para armazenar “bloco” e o “índice” do mesmo.

2.2 FLUXOGRAMA

A execução do programa é bastante simples, se dá pelo seguinte fluxograma.



2.3 COMPILAÇÃO E AMOSTRA DE USO

O programa foi desenvolvido em sistema Linux, e compilado conforme a imagem abaixo.

```

Workspaces  Aplicativos  27 de set 16:26
lucas@pop-os: ~/cacheSim
lucas@pop-os:~/cacheSim$ ls
algoritmo.py  cache.c  cache.h  cacheSim  cacheSim-Gabi-Lucas.tar.gz  compilacao.png  entradas  main.c
lucas@pop-os:~/cacheSim$ rm cacheSim
lucas@pop-os:~/cacheSim$ gcc -o cacheSim -I. main.c -lm -Wall -Werror
lucas@pop-os:~/cacheSim$ ./cacheSim /home/lucas/cacheSim/entradas/1.in
16 16 1 FIFO WB WA 1 300 500
2404
0 3
1
10057 10061 10062 NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL
lucas@pop-os:~/cacheSim$
lucas@pop-os:~/cacheSim$
lucas@pop-os:~/cacheSim$ ./cacheSim /home/lucas/cacheSim/entradas/12.in
256 16 1 FIFO WB WA 4 100 300
24596
948 52
16
0 48 32 16 NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL
33 1 17 49 NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL
2 50 34 18 NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL
3 19 35 NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL
4 36 20 NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL
21 37 5 NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL
6 22 38 NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL
39 23 7 NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL
40 8 24 NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL
41 9 25 NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL
10 42 26 NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL
43 11 27 NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL
28 12 44 NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL
29 13 45 NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL
14 46 30 NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL
31 15 47 NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL NULL
lucas@pop-os:~/cacheSim$
lucas@pop-os:~/cacheSim$
lucas@pop-os:~/cacheSim$
lucas@pop-os:~/cacheSim$

```

3 CONCLUSÃO

O programa executou conforme esperadom, não houve problemas para compilar no sistema em que foi feito e todos os “misses” e “hits”, bem como a posição e alocação dos dados na memória cache tiveram perfeita acurácia conforme os resultados modelos apresentados pelo orientador. Porém, quanto ao tempo total, houve dificuldade em obter os valores idênticos ao modelo.