Organização e Arquitetura de Computadores - Trabalho 3

João Paulo Medeiros Cecilio Bruno Bragança Mendes

Novembro de 2017

1 Introdução

Inicialmente colocamos todas instruções do programa em um arquivo excel, inserindo em colunas cada um dos acesos à memória já convertidos para binário.

Após isso, fizemos uma lista de todos esses acessos e salvamos no arquivo *acessos.txt*, carregando esse arquivo em um script Python para realização dos testes e estatíticas. Abaixo arquivos utilizados no trabalho.

1. codigoacessos.xlsx - Arquivo usado para transcrever os endereços de memória
2. acessos.txt - Analítico de todos os endereços de memória usados
3. associativa.py - Script python para os testes de mapeamento associativo

2 Testes com Mapeamento Direto

3 Testes com Mapeamento Associativo

Para realização dos testes de mapeamento direto criamos as seguintes estruturas de dados: uma lista de listas (endereco a ser acesso e resultado de acerto/erro), outra lista para representar a memória associativa, um dicionário para controlar o número de vezes que cada tag foi usada e um outro dicionário representando o conteúdo da cache.

No início do script, quem estiver executando pode selecionar cada uma das 3 opções de acordo com o tamanho da tag e da palavra, e logo após verificar os resultados.

Utilizamos a política de substituição LFU (Least Frequent Used), que é controlada pelo dicionário já mencionado.

Abaixo trecho de código principal, que faz a simulação do mapeamento associativo.

Listing 1: Loop principal da gestão de acesso

```
for i in lista_acessos:
    # Inicia percorrendo cada endereço,
    # extrai os primeiros x carcteres do endereço e grava na variável tag.
    # Se a tag for encontrada na lista que corresponde a mem. associativa,
    # grava na lista HIT, adiciona o controle de uso e grava todo end. na cache
    tag = i[0][:tam_tag]
    if tag in lista_mem_associativa:
        i[1] = 'HIT'
        dic_cache_controle_uso[tag] = dic_cache_controle_uso[tag] + 1
        cache[tag].add(i[0])

    # Se a tag nao for encontrada na memoria associativa, grava MISS na lista
    # de acessos, adiciona ela, se tiver espaco livre, na memoria associativa
    # depois atualiza controle de uso e cache
    else:
        i[1] = 'MISS'
        if(len(lista_mem_associativa)<tam_cache):
            lista_mem_associativa.append(tag)
            dic_cache_controle_uso[tag] = 1
            cache[tag] = set()
            cache[tag].add(i[0])

        # Nao tendo espaco na mem. associativa, ordena o dicionario de uso
        # do menor uso para o maior, pegando a menos usada para ser substituida
        # retira a tag menos, usada, coloca a nova, e atualiza todas estruturas
        # usada.
        else:
            for key in sorted(dic_cache_controle_uso):
                tag_menos_usada = key
                lista_mem_associativa[lista_mem_associativa.index(
                tag_menos_usada)] = tag
                dic_cache_controle_uso.pop(tag_menos_usada)
                dic_cache_controle_uso[tag] = 1
                cache.pop(tag_menos_usada)
                cache[tag] = set()
                cache[tag].add(i[0])
                break
```

3.1 Conceitos básicos em XXX

3.1.1 Conceitos básicos em XXX.1

Aqui vai um exemplo de uma lista numerada

1. codigoacessos.xlsx -
2. segunda característica desta fase
3. terceira característica desta fase

3.1.2 Conceitos básicos em XXX.2

Seguem-se algumas definições fundamentais para se perceberem as ideias defendidas a seguir:

conceito 1 respectiva definição 1

conceito 2 descrição do conceito 2

3.2 Conceitos básicos em YYY

4 Conclusão

Síntese do que foi dito.
Lista dos resultados atingidos:

- resultado 1
- resultado 2

Conclusão final e Trabalho Futuro.