Organização e Arquitetura de Computadores (ES) - Trabalho Prático 3

Prof. Sergio Johann Filho

Enunciado: Considere o programa em linguagem de montagem abaixo. São executadas 77 instruções, e o número total de referências à memória é maior do que isso (instruções + dados).

```
; Bubble Sort

.code
  lda tam
  add #-1
  sta tmp1
main_loop:
  lda #vet
  add tmp1
  sta endvar1
  lda endvar1,I
  lda tmp1
  add #-1
  sta tmp2
loop:
  lda #vet
  add tmp2
  sta endvar2
  lda endvar2,I
  not
  add #1
  add endvar1,I
  jn troca
  jmp nao_troca
troca:
  lda endvar1,I
  sta var1
  lda endvar2,I
  sta endvar1,I
  lda var1
  sta endvar2,I
nao_troca:
  lda tmp2
  add #-1
  sta tmp2
  jn sai
  jmp loop
sai:
  lda tmp1
  add #-1
  sta tmp1
  jz fim
  jmp main_loop
fim:
  hlt
.endcode


org #90h
.data
tmp1: db #00  ; contador 1
tmp2: db #00  ; contador 2
var1: db #00  ; conteudo da
    pos endvar1
endvar1: db #00 ; posicao 1
    no vetor
endvar2: db #00 ; posicao 2
    no vetor
tam: db #03 ; tamanho do
    vetor
vet: db #34h,#11h,#27h
.enddata
```

Execute o programa no simulador. Quando a instrução não referencia variável, observe apenas os endereços do código (a instrução está marcada no simulador) - um ou dois acessos à memória, dependendo da instrução. Quando a instrução referencia variável, observe os endereços do código e endereço da variável na tabela de símbolos - 3 acessos à memória (no modo direto) e 4 acessos no modo indireto. Por exemplo, a execução das 7 primeiras instruções do programa gera os seguintes endereços: 0x00, 0x01, 0x95, 0x02, 0x03, 0x04, 0x05, 0x90, 0x06, 0x07, 0x08, 0x09, 0x90, 0x0a, 0x0b, 0x93, 0x0c, 0x0d, 0x93, 0x98. O seu trabalho consiste em:

- Derivar o conjunto de endereços gerado pelo programa (todas referências à memória);
- Criar a sequência de acertos / erros em um sistema com *cache* em seis configurações diferentes (espaço de endereçamento de 8 bits):
  - Mapeamento direto, com 3 bits para tag, 2 bits para linha e 3 bits para palavra (cache com 4 linhas, 8 palavras por linha).
  - Mapeamento direto, com 3 bits para tag, 3 bits para linha e 2 bits para palavra (cache com 8 linhas, 4 palavras por linha).
  - Mapeamento direto, com 3 bits para tag, 4 bits para linha e 1 bit para palavra (cache com 16 linhas, 2 palavras por linha).
  - Mapeamento associativo, com 5 bits para tag e 3 bits para palavra (cache com 4 linhas, 8 palavras por linha).
  - Mapeamento associativo, com 6 bits para tag e 2 bits para palavra (cache com 8 linhas, 4 palavras por linha).
  - Mapeamento associativo, com 7 bits para tag e 1 bit para palavra (cache com 16 linhas, 2 palavras por linha).
- Escrever um relatório contendo a sequência de endereços gerados, acertos e erros (nas seis configurações, inclua também o percentual de acertos) e o conteúdo da cache no final da execução para as seis configurações (incluindo tag ou memória associativa). No caso de memória associativa, descreva a política de substituição utilizada.

O conteúdo da *cache* deve ser representado pelo próprio endereço, ou seja, não é necessário colocar instruções ou dados nas linhas. Dessa forma, assuma que cada palavra possui 8 bits.